

Oracle® Cloud

Integrating and Extending Oracle Content Management



F24100-76
September 2023



Oracle Cloud Integrating and Extending Oracle Content Management,

F24100-76

Copyright © 2017, 2023, Oracle and/or its affiliates.

Primary Author: Clare Yan

Contributors: Bruce Silver, Bonnie Vaughan, Sarah Bernau, Promila Chitkara, Kannan Appachi, Robert Briggs, David Jones, Bob Lies, Keith MacDonald, Mark Paterson, Angelo Santagata, Ankur Saxena, Keith Sholes, Ron van de Crommert, Archana Vishnu, Igor Polyakov

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xii
Documentation Accessibility	xii
Diversity and Inclusion	xii
Related Resources	xii
Conventions	xiii

Part I Introduction

1 Get Started

Understand Integrations	1-1
Integration Interfaces	1-4

Part II Enabling Oracle Content Management Integrations

2 Integrate with Other Oracle Applications and Services

Integrate with Oracle Business Intelligence Publisher	2-2
Integrate with Oracle Cobrowse Cloud Service	2-4
Integrate with Oracle Commerce	2-5
Integrate with Oracle Developer Cloud Service	2-7
Integrate with Oracle Eloqua	2-7
Choose an Asset Repository and Create a Publishing Channel	2-8
Provide Oracle Content Management Information for the Eloqua Integration	2-9
Enable Oracle Content Management Embedded Content	2-10
Use an Asset in an Eloqua Landing Page	2-10
Integrate with Oracle Enterprise Contracts	2-12
Integrate with Oracle Integration	2-14
Configure Oracle Integration Settings in Oracle Content Management	2-16
Oracle Integration with Assets	2-18

Oracle Integration with Documents	2-22
Oracle Integration with Sites	2-25
Pass a CSS Style Sheet to Oracle Integration	2-25
Start the Default Version of an Oracle Integration Process	2-25
Integrate with Oracle Intelligent Advisor	2-26
Integrate with Oracle JD Edwards	2-27
Integrate with Oracle Logistics Cloud	2-28
Integrate with Oracle Maxymiser	2-29
Integrate with Oracle Responsys	2-30
Choose an Asset Repository and Create Two Publishing Channels	2-31
Enable the Integration	2-31
Create and Publish Assets in Oracle Content Management	2-32
View Images in Responsys Message Preview	2-34
Integrate with Oracle Visual Builder	2-35
Use Oracle Content Management Components in Oracle Visual Builder Applications	2-37
Embed Oracle Visual Builder Applications in an Oracle Content Management Site Page	2-39
Embed a VBCS Visual App in an Oracle Content Management Page	2-40
Embed a VBCS Page in a Site Page	2-43
Build an Oracle Content Management VBCS Form and Data Report Components	2-48
Build an Oracle Content Management VBCS Public Form Component	2-48
Build an Oracle Content Management VBCS Secure Form Component	2-56
Build an Oracle Content Management VBCS Public Gated Form Component	2-66
Build an Oracle Content and Experience VBCS Data Report Component	2-76
Build an Oracle Content Management VBCS Multipage Form As a Web App Component	2-80
Provide a VBCS Endpoint As a URL for Select Menus	2-87
Integrate with Oracle WebCenter Content	2-89

3 Integrate with Third-Party Applications

Integrate with Desygner	3-1
Add Desygner as SAML Application in Oracle Identity Cloud Service	3-2
Create an Application User for Desygner	3-7
Enable Desygner Integration in Oracle Content Management	3-9
Work with Desygner Asset Types	3-12
Integrate with Kaltura Video Management	3-13
Integrate with Microsoft Office	3-14
Add Oracle Content as a Save/Open Location in Microsoft Office Applications	3-15
Access Oracle Content Features Within Microsoft Office Applications	3-15
Add Links to Cloud Items Directly from Microsoft Outlook	3-16
Create or Edit Microsoft Office Files from the Oracle Content Management Web Interface	3-18

Integrate with Slack	3-19
Enable and Configure Integration with Slack in Oracle Content Management	3-20
Create an App for Slack Using the Slack Website and Install	3-21
Add the App Credentials to Oracle Content Management	3-21

4 Use Content Connectors

Enable a Content Connector	4-2
Disable a Content Connector	4-3
Configure a Google Drive Content Connector	4-3
Configure a Microsoft OneDrive Content Connector	4-6
Configure a Dropbox Content Connector	4-8
Enable Jax-WS	4-10
Configure a WordPress.org Content Connector	4-10
Configure a YouTube Content Connector	4-11
Configure a Microsoft SharePoint Online Content Connector	4-13
Configure a Contentful Content Connector	4-15
Configure a Drupal Content Connector	4-16
Configure Oracle WebCenter Content Server and Oracle Content Management for the WCC Connector v2.0	4-17
Verify Network Accessibility for a WCC Connector v2.0	4-18
Check WebCenter Content Server Readiness	4-18
Enable SSL	4-18
Enable Jax-WS	4-19
Set Up a Security Policy	4-19
Run the Indexer and File Formats Wizard	4-23
Check Oracle Content Management Readiness	4-24
Configure the Oracle WebCenter Content Connector v2.0	4-24
Specify WebCenter Content Connector v2.0 Mappings	4-25
Enable Oracle WebCenter Content Connector v2.0 for an Asset Repository	4-26
Use Oracle WebCenter Content Connector v2.0	4-27
Map to a Custom Asset Type	4-29
Revoke Authorized Users	4-30
Use Custom Digital Asset Types in Content Connectors	4-30
Import Assets Mapped to Digital Asset Types	4-30
Create and Configure a Custom Content Connector	4-31
Create Content Types for a Connector	4-32
Map Source Metadata to Fields in a Content Type	4-32
Provide Configuration Parameter Values for a Content Connector	4-33
Delete a Content Connector	4-33

Part III Developing Oracle Content Management Extensions

5 Develop Custom Actions with Application Integration Framework (AIF)

Understand the Application Integration Framework (AIF)	5-1
Manage Custom Applications	5-3
Configuration File Format	5-4
Application Properties	5-7
Action Command	5-8
Invoke Command	5-9
Presentation Command	5-9
Expressions	5-10
Variables	5-11
Localization	5-14

6 Develop Content Connectors

Connector REST API Interface	6-1
Connector SDK	6-1
Build a New Content Connector	6-2
REST Interfaces for Configuration, Authorization, and Fetching Content	6-4
REST Interfaces for File System Browsing and Searching	6-9
Content Picker	6-11
Authorization	6-17
Content Connector Configuration and Registration	6-21
Content Connector Execution Flow	6-21
Pexels Content Connector Sample Implementation	6-22
Install the Content Connector	6-22
Check Prerequisites for Installation	6-22
Build the Content Connector WAR File	6-23
Register the Content Connector	6-23
Test the Content Connector	6-24
Understand the Content Connector Source Code	6-25
Custom Picker UI	6-25
Pexels REST APIs	6-26
Change and Test the Content Connector Code	6-26
Download the CEC Content Connector Sample and SDK	6-26

7 Develop Custom Field Editors

Create a Custom Field Editor	7-1
------------------------------	-----

Configure Content Type to Use Custom Field Editor	7-2
Edit a Custom Field Editor	7-2
appinfo.json for Custom Field Editors	7-3
edit.html for Custom Field Editor	7-6
view.html for Custom Field Editor	7-9
Sample Content Field Editors	7-10
Slider	7-11
Location Selector	7-11
Content Field Editor SDK Reference	7-11
Content Field Editor SDK Object	7-12
getField()	7-12
getFields()	7-12
getLocale()	7-13
getSetting(setting)	7-13
getSettings()	7-14
registerDisable(callback)	7-14
setValidation(callback)	7-15
resize(size)	7-15
openContentPicker()	7-15
getDirection()	7-16
Custom Content Field Object	7-16
getName()	7-17
getDefaultValue()	7-17
getDataType()	7-17
setValue(value)	7-18
getValue()	7-18
on(event, callback)	7-18

8 Develop Custom Content Forms

Create a Custom Content Form	8-1
Configure a Content Type to Use a Custom Content Form	8-1
Edit a Custom Content Form	8-2
appinfo.json	8-2
edit.html for Custom Content Forms	8-4
Content Form SDK Reference	8-8
Custom Content Form SDK Object	8-8
Custom Content Form Type Object	8-15
Custom Content Form Item Object	8-16
Form Options	8-29
Custom Content Form Field Object	8-31

CustomEditor Object	8-39
Sample Custom Form	8-41
Get Custom Form Sample	8-42
Add OCM Image Picker and Link Dialog Plug-ins for Rich Text Editor	8-42

9 Develop Translation Connectors for Language Service Providers

Overview of the Translation Connector Framework	9-2
Translation Connector SDK	9-2
Translation Connector REST APIs	9-3
Request a Lingotek Trial Connector for Content Translation	9-4
Enable a Lingotek LSP Translation Connector	9-7
Delete a Lingotek LSP Translation Connector	9-7
Register Multiple Lingotek Connectors	9-8
Add Custom Locales to a Lingotek Translation Connector	9-10
Translate Native Files in Assets	9-12
Build a New Translation Connector	9-12
REST Interfaces for Configuration and Authorization	9-13
REST Interfaces for Creating Translation Jobs and Returning Translated Content	9-19
Configure and Register a Translation Connector	9-20
Translation Connector Execution Flow	9-21
Translation Job Editor	9-21
Translation Jobs Validation	9-25
Sample Translation Connector Implementation	9-28
Create the Sample Translation Connector with Content Toolkit	9-28
Register the Sample Translation Connector	9-28
Test the Sample Translation Connector	9-29
Understand the Sample Translation Connector Source Code	9-30
Translation Job Original Zip File Format	9-32
Translation Job Translated Zip File Format	9-33

10 Develop External Processors

External Processor Execution Flow	10-1
Pull Model	10-2
REST API for Content Capture	10-2
External Processor SDK	10-3
External Processor Examples	10-3

11 Compile Content Layouts as HTML

Part IV Developing Oracle Content Management Integrations

12 Understand Cross-Origin Resource Sharing (CORS)

13 Embed the Web User Interface in Other Applications

14 Oracle Content Management REST APIs

Integrate with Oracle Content Management Using OAuth	14-2
Cloud Account Using IAM Identity Domain	14-4
Access OCM Using Client Credentials (Two-Legged OAuth in Identity Domain)	14-4
Access OCM Using Authorization Code (Three-Legged OAuth Flow in Identity Domain)	14-7
Access OCM Using Resource Owner (Identity Domain)	14-11
Cloud Account Using Oracle Identity Cloud Service	14-14
Access OCM Using Client Credentials (Two-Legged OAuth Flow)	14-14
Access OCM Using Authorization Code (Three-Legged OAuth Flow)	14-17
Access OCM Using Resource Owner	14-21
Download the Swagger File for a REST API	14-24
REST API for Activity Log	14-25
REST API for Content Capture	14-25
REST API for Content Delivery	14-26
REST API for Content Management	14-27
REST API for Content Preview	14-29
REST API for Conversations	14-30
REST API for Documents	14-30
REST API for Self-Management	14-31
REST API for Sites Management	14-31
REST API for Users and Groups	14-32
REST API for Webhooks Management	14-32
Use REST APIs for Content Search	14-32
Search Query Operators	14-32
Search Queries	14-39
Supported Date and Time Formats	14-42

Search with the Querytext Parameter	14-43
Set Up Searches on Metadata Fields	14-44
Search Request Parameters	14-44
Two-Level Deep Search	14-57
Search JSON Data in JSON Fields	14-58
Search Across Types	14-59
The fields Parameter	14-60
The orderBy Parameter	14-60
Dynamic Count of Assets per Taxonomy Category	14-60
The AGGS Query Parameter	14-61
Aggregation Cache	14-63
Using Dynamic Asset Counts per Category API	14-63
An e-commerce Use Case	14-64
A General Use Case	14-78
Scroll API	14-82
Custom Ranking Policies	14-83
Built-in Ranking Policy	14-83
Custom Ranking Policies	14-84
Custom Ranking Policies Lifecycle	14-84
Supported Ranking Methods	14-85
Use REST APIs for Extended Workflow	14-94
Complete Workflow Instance API	14-94
Search In-Workflow Assets	14-95
Create and Use Applinks for File and Folder Access	14-96
Provide Access to Files and Folders with Public Links	14-97
Upload a REST API Swagger File into Mobile Cloud Service	14-99

15 Oracle Content Management APIs

Embed UI API V2	15-1
Site Compile API	15-1
Sites Component API	15-1
Sites Rendering API	15-1

16 Oracle Content Management SDKs

Content SDK for JavaScript	16-1
Content SDK for Java	16-1
Content SDK for Swift	16-2
Sites SDK	16-2

17 GraphQL Support in Oracle Content Management

Get Started with GraphQL	17-1
GraphQL Schema	17-3
GraphQL Queries	17-5
GraphQL Support for Content Preview	17-14
GraphQL Samples	17-14

18 Use Webhooks

Outgoing Webhooks	18-1
Configure Outgoing Webhooks	18-1
Monitor Webhook Events	18-3
Use Endpoints for Push Notifications	18-6
Receive Push Notifications from an Asset Lifecycle Webhook	18-7
Receive Push Notifications from an Asset Publishing Webhook	18-8
Receive Push Notifications from a Site Publishing Webhook	18-9
Receive Push Notifications from a Prerender Webhook	18-9
Receive Push Notifications from a Scheduled Jobs Webhook	18-9
Incoming Webhooks	18-9
Configure an Incoming Webhook	18-10
Use the CAL-Based Webhook Adapter	18-11

19 Set Proxies

Configure Proxy Service Settings	19-1
Add Logged User Data to a Request Through a Secure Proxy Endpoint	19-3
Debug Proxy Service Endpoints	19-4

Preface

Integrating and Extending Oracle Content Management describes how to configure Oracle Content Management and combine it with other services to create custom integrations and to extend the capabilities of Oracle Content Management.

Audience

This publication is intended for Oracle Content Management administrators and developers who want information about integrating Oracle Content Management with other services to provide custom applications.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

For more information, see these Oracle resources:

- *Getting Started with Oracle Cloud*
- *Administering Oracle Content Management*
- *Building Sites with Oracle Content Management*

- *Collaborating on Documents with Oracle Content Management*
- *Developing with Oracle Content Management As a Headless CMS*
- *Managing Assets with Oracle Content Management*
- *Building Sites with Oracle Content Management*
- *What's New for Oracle Content Management*
- *Known Issues for Oracle Content Management*

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Introduction

Oracle Content Management (OCM) is a cloud-based content hub to drive omni-channel content management and accelerate experience delivery. It offers powerful collaboration, workflow management, and development capabilities to streamline the creation and delivery of content and improve customer and employee engagement.

As an Oracle Platform-as-a-Service (PaaS), Oracle Content Management works seamlessly with other Oracle Cloud services and applications. The integration features are key components in several Oracle offerings and make it easy for you to leverage the service in your own applications.

[Guided tour](#)

These are some prerequisites if you want to integrate and extend Oracle Content Management:

- An active Oracle Cloud account with an OCM instance. See [Deploy OCM in a Region with Identity Domains](#).

Note:

Oracle is merging the capabilities of Oracle Identity Cloud Service (IDCS) into the native Oracle Cloud Infrastructure Identity and Access Management (OCI IAM) service. All IDCS features and functionality will continue to exist as part of OCI IAM. If your region hasn't been updated, follow the steps in [Deploy OCM in a Region without Identity Domains](#).

- You need to be assigned with the right access role for integrations. See [Typical Organization Roles](#).

Note:

To access integration features, you need to be assigned with the right role. There are different types of roles in Oracle Content Management. Understanding how roles work is essential for accessing integration features.

- Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM.

Explore this part further to [get started](#) with integrating and extending with Oracle Content Management.

1

Get Started

The following topics gives you an overview of Oracle Content Management integrations with other applications and services:

- [Understand Integrations](#)
- [Integration Interfaces](#)

Understand Integrations

Oracle Content Management provides multiple ways to integrate its functionality, whether you want to incorporate your processes or apps into Oracle Content Management, or simply use Oracle Content Management in your enterprise application.

Oracle Content Management

Oracle Content Management provides rich content management features, from folder and file viewing and sharing, to conversations, to websites that deliver your message and content securely.

- Integrations with JD Edwards, Oracle Business Intelligence, and other services show that Oracle Content Management is a key component in a number of Oracle integrations.
- An embeddable version of the web user interface and website components for interacting with folders, files, conversations, and processes provide ready-to-use integrations.
- [Oracle Content Management REST APIs](#) and the Oracle Content Management SDKs let you access Oracle Content Management functionality to create your own integrations within the service or across services.
- Single Sign-On (SSO) authentication provides a seamless user experience across services.

Available Integrations


Oracle Content Management is a key component in a number of Oracle integrations. With some integrations, Oracle Content Management is provided “out of the box” as part of the service. For others, you must enable or configure the integration.



Note:

A number of the integrations described in this guide require that integrated services be in the same identity domain. For that reason, those integrations work only on traditional cloud accounts.

Category	Integrations
Middleware	<p>Oracle WebCenter Content: Use Oracle Content Management to provide a truly comprehensive hybrid enterprise content management (ECM) integration, with a unified ECM infrastructure and security from a single vendor. It combines anywhere access from the cloud with content retention and archiving from on-premise installations.</p>
Applications	<p>Oracle JD Edwards: Use Oracle Content Management to attach managed documents to transactions and collaborate through conversations.</p>
Software as a Service (SaaS)	<p>Oracle Cobrowse Cloud Service: Use Oracle Content Management to work with a site and add the Cobrowse Launcher component to a site page.</p> <p>Oracle Commerce: Use Oracle Content Management to enhance content collaboration and streamline content creation and publication for commerce. For example, once the integration is enabled, marketing content such as blogs and digital assets from Oracle Content Management repositories can be pulled into Oracle Commerce to be rendered both statically and dynamically.</p> <p>Oracle Eloqua: Use Oracle Content Management to enable Eloqua users to insert published images from Oracle Content Management's asset repository into Eloqua responsive emails and landing pages. The integration also allows users to upload images to Oracle Content Management directly from within Eloqua. This allows Eloqua users to leverage Oracle Content Management's extensive asset repository capabilities to store content, while using Eloqua to design marketing assets.</p> <p>Oracle Enterprise Contracts: Use Oracle Content Management to collaborate on contract negotiations. The integration of OCM with Oracle Enterprise Contracts allows stakeholders to share contracts for review and revision, collaborate and edit contracts in Microsoft Word from the OCM UI.</p> <p>Oracle Intelligent Advisor: Use Oracle Content Management to add an Intelligent Advisor component to site pages.</p> <p>Oracle Logistics Cloud: Use Oracle Content Management to store and manage documents.</p> <p>Oracle Maxymiser: Use Oracle Content Management's extensive asset repository capabilities to store content, while using Maxymiser to design campaigns.</p> <p>Oracle Responsys: Use Oracle Content Management to enable Responsys users to insert content assets from Oracle Content Management's asset repository into Responsys Email and Mobile campaigns. This allows Responsys users to leverage Oracle Content Management's extensive asset repository capabilities to store content, while using Responsys to design campaigns.</p>

Category	Integrations
Platform as a Service (PaaS)	<p>Oracle Business Intelligence Publisher: Use Oracle Content Management for managed folders as a destination for generated reports.</p> <p>Oracle Developer Cloud Service: Use project templates and tools to create, test, and package your own site templates, themes, and components for use in Oracle Content Management.</p>
	<div style="border: 1px solid #0070C0; padding: 10px; background-color: #E6F2FF;"> <p> Note:</p> <p>This integration is available with traditional cloud accounts.</p> </div>
	<p>Oracle Integration, Oracle Process Cloud Service: Automate business-driven, company-specific processes, such as employee onboarding or IT service requests, and incorporate those processes into Oracle Content Management.</p> <p>Oracle Visual Builder: Rapidly create web and mobile applications with minimal to no coding using an open-source, standards-based integration to develop, collaborate on, and deploy applications within Oracle Content Management.</p>
Third-party applications	Oracle Content Management includes integrations for several third-party applications, such as Microsoft Office, Kaltura Video Management - Video Plus, Desygner, and Slack. You just need to enable the integration in your instance. Additionally, Oracle Cloud Marketplace lists applications created by partners using the integration features provided with Oracle Content Management.
Custom applications	Use options such as REST APIs, Java services, and the Application Integration Framework (AIF) to create any number of applications.

Use Apps and Services in Oracle Content Management

If you want to expand the service to include your own apps or to communicate with other services, the following text discusses what you can do:

- The open architecture for site components means you can register and deliver hosted apps and create your own components using your preferred platform. For details about how to create your own components, see [Develop Components](#).
- Cross-Origin Resource Sharing (CORS) allows a web page to make requests such as `XMLHttpRequest` to another domain. If you have a browser application that integrates with Oracle Content Management but is hosted in a different domain, add the browser application domain to Oracle Content Management's CORS origins list. See [Understand Cross-Origin Resource Sharing \(CORS\)](#).
- If you use REST services that do not support Cross-Origin Resource Sharing (CORS) or that require service account credentials, you can use the Oracle Content Management proxy service. See [Configure Proxy Service Settings](#).
- You can use Application Integration Framework (AIF) to create your own custom applications that define the actions that are exposed in the web interface, respond to user selections, call third-party services, and specify how the results are presented to the user. The framework supports variables and expressions and provides multiple language support. See [Understand the Application Integration Framework \(AIF\)](#).

- You can modify the web interface and menus to provide access to your applications and features. See [Manage Custom Applications](#).

Use Oracle Content Management with Other Services

The Oracle Platform as a Service (PaaS) architecture means you can leverage the Oracle Content Management functionality where you need it:

- Provide direct interaction with Oracle Content Management in another web application with the embedded version of the web user interface.
- Specify a list of domains where you allow content from Oracle Content Management to be displayed using either the embedded web user interface or REST calls. See [Embed UI API V2 for Oracle Content Management](#).

Integration Interfaces

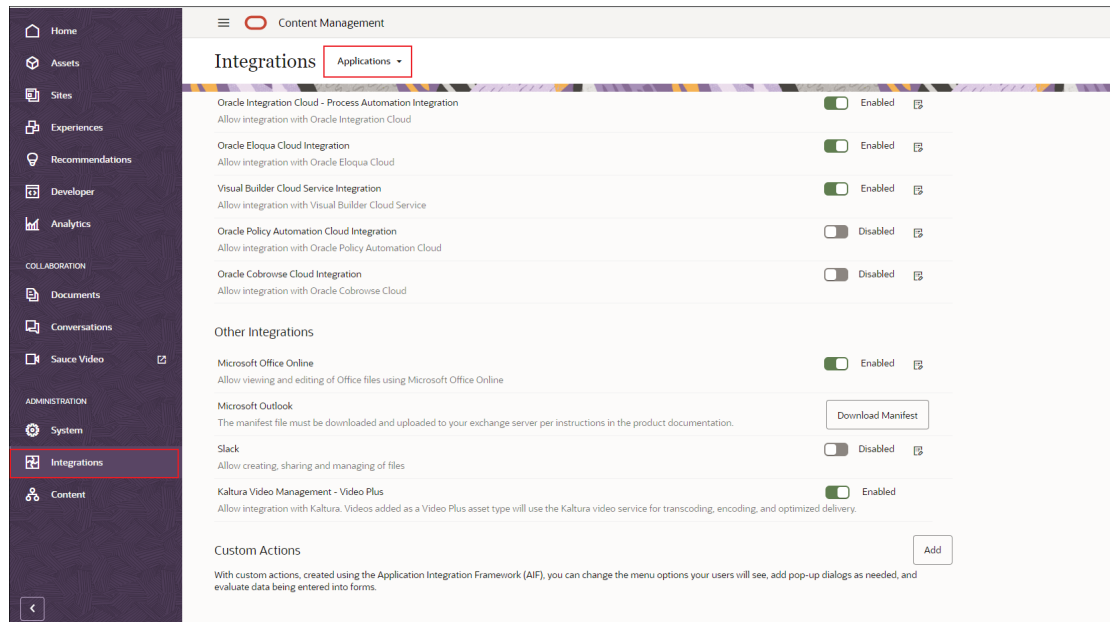
From the **Integrations** page in the Oracle Content Management (OCM) **Administration** web user interface, you can select an application for integration, configure content connectors to third-party content repositories, and configure proxy service settings.

Integrations are configured in various places:

- Some integrations need to be enabled in the OCM web interface on the **Integrations** page before they're available. There may be configuration steps that need to be done on the Oracle Content Management side.
- Other configurations steps may need to be performed on the "other side" (i.e. within the service that OCM is being integrated).

Note:

Whenever one application is to be configured to integrate with another application (i.e. the target application), the user needs to have the target application's server connection information such as username, user credential, and service URL. Depending on the integration scenario, you may have a different set of user credentials and service URL.



You can also add custom actions created with the [Application Integration Framework \(AIF\)](#) to change the menu options for your users, add pop-up dialogs, and evaluate data entered into forms.

This image shows the wide view of the menu on the **Integrations** page. By default, this page shows the narrow view, with abbreviated menu items. To switch between narrow and wide views of the menu, click the arrow at the bottom left of the page.

Learn more about integrations in the video [Integrations in Oracle Content Management](#).

Part II

Enabling Oracle Content Management Integrations

This part focuses on enabling Oracle Content Management (OCM) integrations so you can begin integrating with other applications and services.

While some integrations with Oracle applications are enabled from the Oracle Content Management web interface, some integrations are enabled from other Oracle applications and services before integration functionalities can be utilized.

You can use Oracle Content Management's content connectors to connect to third-party repositories like Dropbox, Google Drive, or WordPress, and import content from the repository into Oracle Content Management.

Explore more on the topics in the following chapters:

- [Integrate with Other Oracle Applications and Services](#)
- [Integrate with Third-party Applications](#)
- [Use Content Connectors](#)

2

Integrate with Other Oracle Applications and Services

You can integrate Oracle Content Management (OCM) with many Oracle applications and services, as well as third-party applications. Generally, configurations are needed on both systems that are being integrated. Please refer to the specific product for detailed integration instructions.

The following is a full list of applications and services that are available for OCM integration:

Oracle Business Intelligence Publisher	An enterprise reporting solution for authoring, managing, and delivering all your highly formatted documents, such as operational reports, electronic funds transfer documents, government PDF forms, shipping labels, checks, sales and marketing letters, and much more.
Oracle Cobrowse Cloud Service	A collaboration tool for providing online customers on any device with real-time assisted service or guidance.
Oracle Commerce	An ecommerce platform that helps B2C and B2B businesses connect customer and sales data from their CRM to their financial and operational data so they can offer personalized experiences to buyers across sales channels.
Oracle Developer Cloud Service	A cloud-based software development Platform as a Service (PaaS) and a hosted environment for your application development infrastructure. It provides an open-source standards-based solution to plan, develop, collaborate, build, and deploy applications in Oracle Cloud.
Oracle Eloqua	A best-in-class B2B marketing automation solution.
Oracle Enterprise Contracts	A comprehensive offering that standardizes corporate contract policies, improves internal controls, and enforces compliance with all contractual obligations and regulatory requirements. It provides a complete solution for managing sales, procurement, and other contracts.
Oracle Integration	A cloud service that connects any application and data source to automate end-to-end processes and centralize management.
Oracle Intelligent Adviser	A decision automation solution (formerly Oracle Policy Automation) that empowers business users to configure rules, ensuring customers, service agents, and employees always receive enriched, personalized, and precise customer service.
Oracle JD Edwards	A resource planning software that combines business value, standards-based technology, and deep industry experience into a business solution with a low total cost of ownership.

Oracle Logistic Cloud	An integrated solution that seamlessly manages transportation, global trade, and warehouse management activities via unified business processes.
Oracle Maxymiser	A testing and optimization solution that provides advanced website testing, real-time behavioral targeting, in-session personalization, and product recommendations across websites and mobile apps. The solution also offers real-time personalization for B2C and B2B marketing campaigns to increase speed to market and improve customer experiences.
Oracle Responsys	A cross-channel campaign management platform that delivers advanced intelligence at scale so you can create personalized messages based on the individual interests and preferences of customers and prospects.
Oracle Visual Builder	A cloud-based software development Platform as a Service (PaaS) and a hosted environment for your application development infrastructure.
Oracle WebCenter Content	A solution that provides a unified application for several different kinds of content management.
Third-party applications	External applications that are created by partners using the integration features provided with Oracle Content Management.

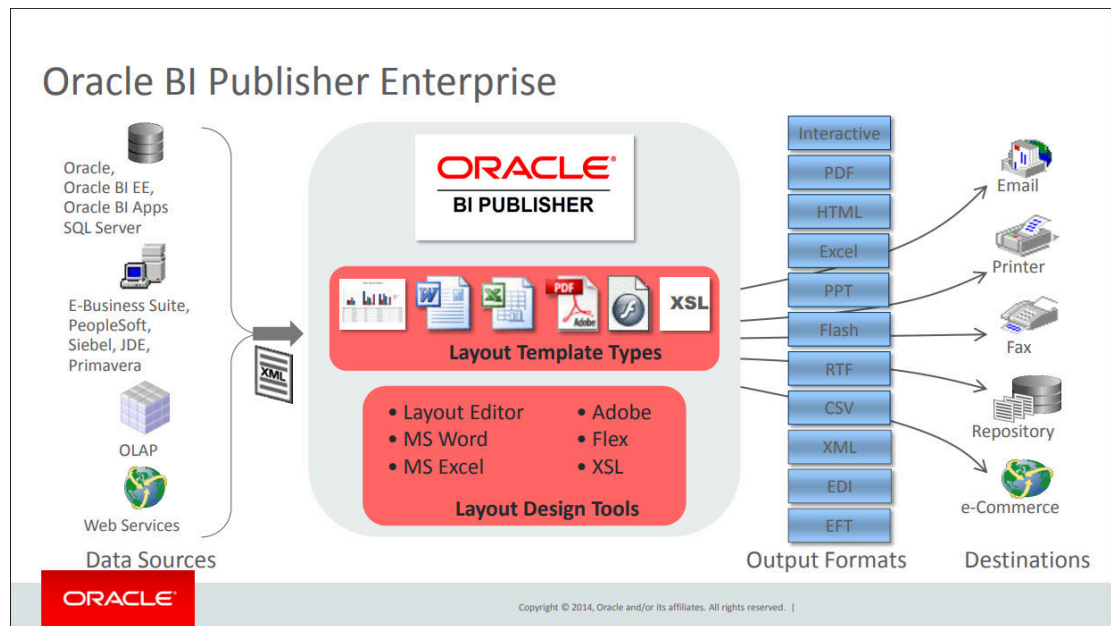
Integrate with Oracle Business Intelligence Publisher

Why Integrate with Oracle Business Intelligence Publisher?

[Oracle Business Intelligence Publisher](#) belongs to [Oracle Fusion Middleware](#), a comprehensive family of software products that includes a range of application development tools and services.

The integration of Oracle Content Management with Business Intelligence Publisher offers managed folders as a destination for generated reports. For example, payroll reports can be sent to individual employee folders. Manage the folders and folder contents with the easy-to-use interface from mobile, web, or desktop devices. All of the familiar security, sharing, viewing, and collaboration features of Oracle Content Management are available to view and manage your reports.

The screenshot below shows you how reports can be generated in multiple output formats and sent to different destinations. A repository is one of the destinations.



Prerequisites

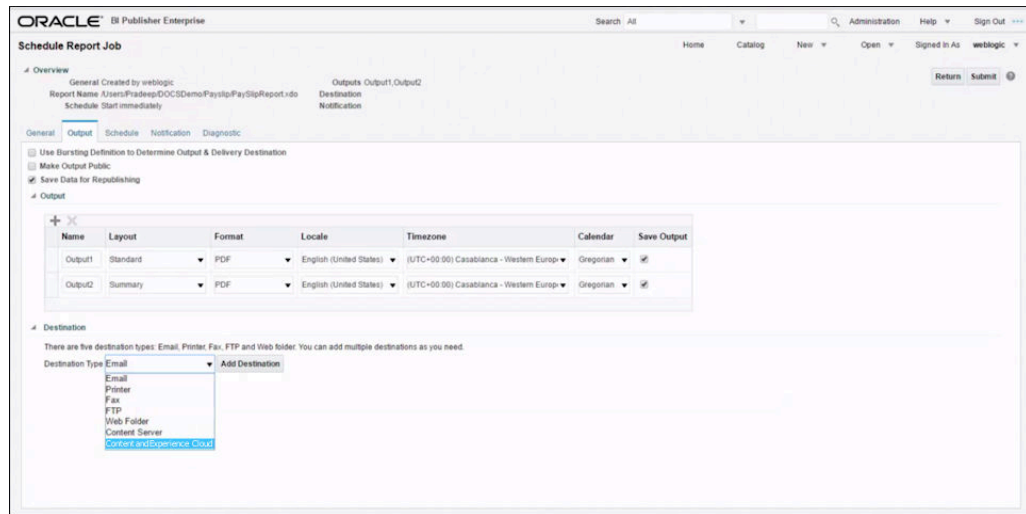
There are prerequisites to integrating Oracle Content Management with Oracle Business Intelligence Publisher. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. See the [Oracle Business Intelligence Publisher documentation](#) for any specific requirement. On the Oracle Content Management side, do the following:

- Create and activate an Oracle Cloud Account before you begin.
- Review Overview of Oracle Content Management, which provides information on interacting with OCM and concepts on access roles for performing certain tasks.
- Compare the starter vs. the premium edition of Oracle Content Management. Only the premium edition has all the features.
- Create an Oracle Content Management instance.
- Take a guided tour of [repositories in Oracle Content Management](#).
- Create an asset repository. To set this up, you need to have the repository administrator (CECRepositoryAdministrator) role in Oracle Content Management.

Integration Process

See [Set Output Options](#) in *Fusion Middleware User's Guide for Oracle Business Intelligence Publisher* on how to configure the application for integration.

The screenshot below shows you how Oracle Content Management in Business Intelligence Publisher is set as an output destination.



Integrate with Oracle Cobrowse Cloud Service

Why Integrate with Oracle Cobrowse Cloud Service?

Oracle Cobrowse is one of Oracle's Software as a Service (SaaS) applications that lets users share screens or initiate a cobrowsing session with another person. For example, you might want to use cobrowse on a sales site so that a sales representative can help a customer select appropriate products or services on the site.

The integration of Oracle Cobrowse with Oracle Content Management (OCM) lets you add the Cobrowse Launcher component to a site page in OCM.

Prerequisites

There are prerequisites before you begin the integration process. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. Please also see the [Oracle Cobrowse documentation](#). On the Oracle Content Management side you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Only administrators with the *enterprise user* role can enable integration with Oracle Cobrowse Cloud Service. If you aren't an enterprise user, the **Oracle Cobrowse Cloud Integration** option is grayed out in Oracle Content Management.

Integration Process

To integrate with Oracle Cobrowse Cloud Service:

1. After you sign in to the Oracle Content Management web application as an administrator, open your user menu and click **Administration**.
2. In the **Administration** menu, click **Integrations**.
3. Under **Oracle Integrations**, select **Oracle Cobrowse Cloud Integration** to enable the service, and then set these values:

- **Service URL:** Enter the URL for the Cobrowse service. See [Log in to the Agent Console](#) in the *Standalone Cobrowse User Guide* for the link (for example, <https://www.livelook.com>).
- **Service User:** Enter the Oracle Cobrowse Cloud Service administrator user name.
- **Service Password:** Enter the user's password.

After you've configured the integration, Oracle Content Management users can enable cobrowse to work with a site and add the Cobrowse Launcher component to a site page.



Note:

If you later decide to disable cobrowse, you must disable the option on the **Integrations** page and in the site settings for any sites that use cobrowse. If you disable only the option on the **Integrations** page, any sites that use cobrowse will continue to do so, but users won't be able to add new cobrowse functionality.

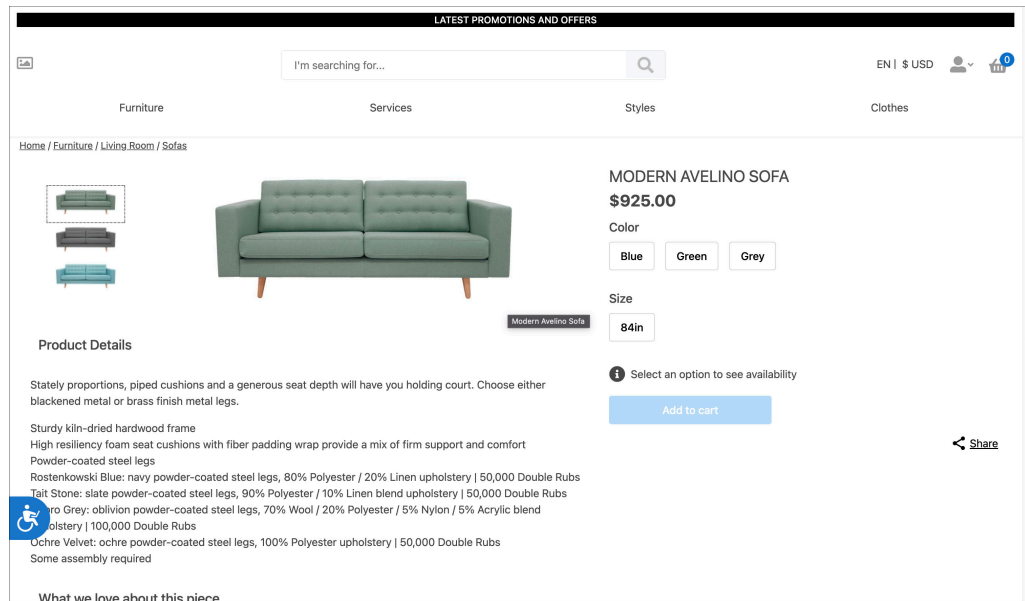
Integrate with Oracle Commerce

Why Integrate with Oracle Commerce?

Oracle Commerce is an ecommerce platform that helps B2C and B2B businesses connect customer and sales data from their CRM to their financial and operational data so they can offer personalized experiences to buyers across sales channels.

The integration of Oracle Content Management (OCM) with Oracle Commerce lets you enhance content collaboration and streamline content creation and publication for commerce. Additionally, the cloud-based content hub promotes content reuse from a single source and accelerates experience delivery.

Once the integration is enabled, for example, marketing content such as blogs and digital assets from Oracle Content Management repositories can be pulled into Oracle Commerce to be rendered both statically and dynamically. In the screenshot below, assets such as product images and texts from Oracle Content Management repositories are published on commerce storefront pages.



Prerequisites

There are prerequisites to integrating Oracle Content Management with Oracle Commerce. On the Oracle Content Management side, you will need the following before the integration process:

- An Oracle Cloud Account.
- An OCM instance.
- A subscription of Oracle Commerce.

Integration Process

On the Oracle Content Management side, do the following:

1. Create a publishing channel and an [asset repository](#). To set this up, make sure you have the right access role, i.e., the repository administrator role (CECRepositoryAdministrator), in Oracle Content Management. Access **Publishing Channels** and **Repositories** in the Content dropdown menu by clicking on the **Content** option in the left navigation menu (under ADMINISTRATION).
 - a. Create a publishing channel.
 - b. Create an asset repository and associate it with the publishing channel you created earlier.
2. Create an integration user and assign these roles.
 - Application roles in Identity Cloud Service: CECEnterpriseUser, CEContentAdministrator, and CECServiceAdministrator define what users can do.
 - Editorial roles in Oracle Content Management: A type of role required for performing tasks with Oracle Content Management repositories and channels that are used in the integration configuration of Oracle Commerce.

 **Note:**

For more details, see [Assign Roles to Groups and Assign Users to Groups](#).

3. Add the integration user as a member to the repository and publishing channel.
4. Publish the assets in the repository to the publishing channel.
5. Complete the integration steps on the other side, see [Oracle Commerce](#) documentation. Note that you need a subscription for the product that you intend to integrate with OCM.

Integrate with Oracle Developer Cloud Service

Why Integrate with Oracle Developer Cloud Service?

[Oracle Developer Cloud Service](#) is a cloud-based software development platform that provides an open source, standards-based integration to develop, test, and deploy applications into other cloud services such as Oracle Content Management.

 **Note:**

This integration works only on [traditional cloud accounts](#) because it requires that the service be in the same identity domain as Oracle Content Management.

Prerequisites

There are prerequisites before you begin the integration process. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Integration Process

With [Oracle Developer Cloud Service](#), you can use [Oracle Content Management Toolkit](#) to develop custom Oracle Content Management applications. Check documentation on both sides to learn about integration details.

Integrate with Oracle Eloqua

Why Integrate with Oracle Eloqua?

[Integration with Eloqua](#) lets you insert published assets from an Oracle Content Management asset repository into Eloqua responsive emails, forms, and landing pages. Using Oracle Content Management as a single source for all images saves rework because you can find your images easily.

Eloqua users can leverage the extensive asset repository capabilities in Oracle Content Management to store content for use in Eloqua marketing assets. When you design emails, forms, or landing pages, this integration gives you the option to insert published image assets from Oracle Content Management into your Eloqua assets.



Prerequisites

These are the prerequisites before you begin the integration process.

Note:

Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. Please also check the [Oracle Eloqua documentation](#).

- Create and activate an Oracle Cloud Account before you begin.
- Review Overview of Oracle Content Management, which provides information on interacting with OCM and concepts on access roles for performing certain tasks.
- Compare the starter vs. the premium edition of Oracle Content Management. Only the premium edition has all the features.
- Create an Oracle Content Management instance.
- Take a guided tour of [repositories in Oracle Content Management](#).
- Create an asset repository. To set this up, you need to have the repository administrator (CECRepositoryAdministrator) role in Oracle Content Management.

Integration Process

The process of Eloqua integrating with Oracle Content Management consists of the following steps:

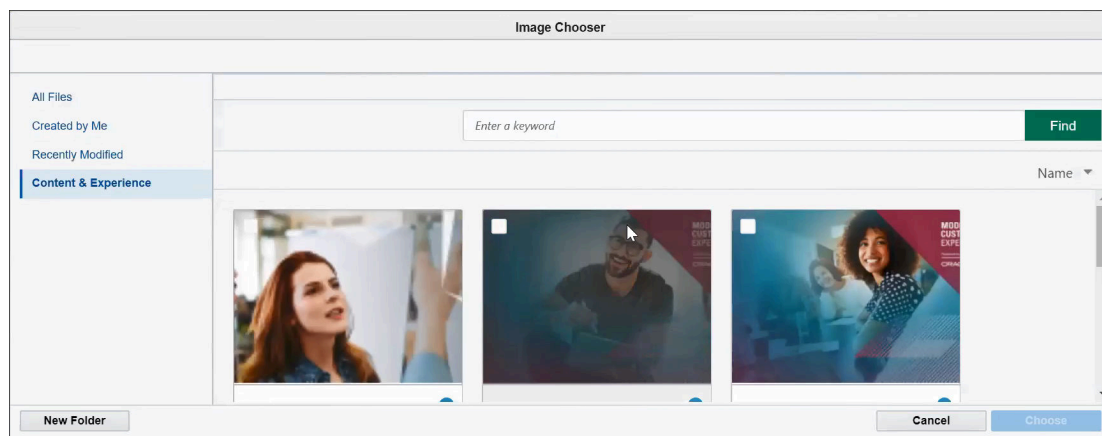
1. [Choose an Asset Repository and Create a Publishing Channel](#)
2. [Provide Oracle Content Management Information for the Eloqua Integration](#)
3. [Enable Oracle Content Management Embedded Content](#)
4. [Use an Asset in an Eloqua Landing Page](#)

Choose an Asset Repository and Create a Publishing Channel

For integration with Eloqua, you need to choose an asset repository and create a publishing channel in Oracle Content Management. Then Eloqua can provide access to assets in the repository.

After you choose an asset repository in Oracle Content Management, create and share a publishing channel. Oracle suggests creating a channel that is specifically for Eloqua.

The assets published to the Eloqua channel will be displayed in a tab on the Eloqua Image Chooser page, where you can choose an Oracle Content Management asset.



To refresh an image in all channels, email campaigns, and landing pages in which the image is used, you can change and republish the image in Oracle Content Management.

If the image you want to use is not available in the Eloqua Image Chooser, you can go to Oracle Content Management and upload or publish the image.

Provide Oracle Content Management Information for the Eloqua Integration

To enable Oracle Content Management integration with Eloqua, submit a service request to My Oracle Support (MOS).

Include the following details for your Oracle Content Management instance:

1. **Domain name**
For example, if you sign in to Oracle Content Management at the URL `https://eloquaoce.oraclecloud.com/documents/assets`, the domain name is `eloquaoce.oraclecloud.com`.
2. **Repository ID**
To retrieve your repository ID:
 - a. In Oracle Content Management, navigate to **Administration > Assets**.
 - b. Select the repository you want to use for storing your Eloqua images.
 - c. The repository ID is appended to the URL. For example, if the full URL is `https://eloquaoce.oraclecloud.com/documents/repository/ABC6F858251160CC3000A497C0C07C96651BA6F0BE73`, the repository ID is `ABC6F858251160CC3000A497C0C07C96651BA6F0BE73`.
3. **Channel ID and channel token**
To retrieve your channel ID and channel token:
 - a. In Oracle Content Management, navigate to **Administration > Assets**.
 - b. From the **Repositories** drop-down menu, choose **Publishing Channels**.
 - c. Select the publishing channel you intend to use for Eloqua images.
 - d. The channel ID and channel Token are listed under *API Information*.
 - e. Ensure that Access is set to **Public**.

Enable Oracle Content Management Embedded Content

In Oracle Content Management administration security settings, enable embedded content and add the Eloqua URL.

Then you can embed the Oracle Content Management web UI in Eloqua. See Embed Content in Other Domains.

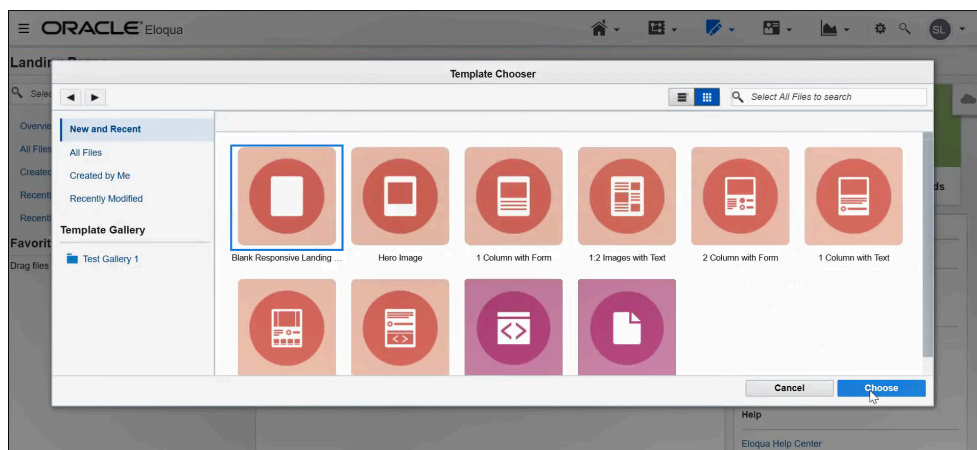
Use an Asset in an Eloqua Landing Page

After the Oracle Content Management Cloud integration with Eloqua is enabled, you can choose a digital asset from Oracle Content Management to use in an Eloqua responsive landing page.

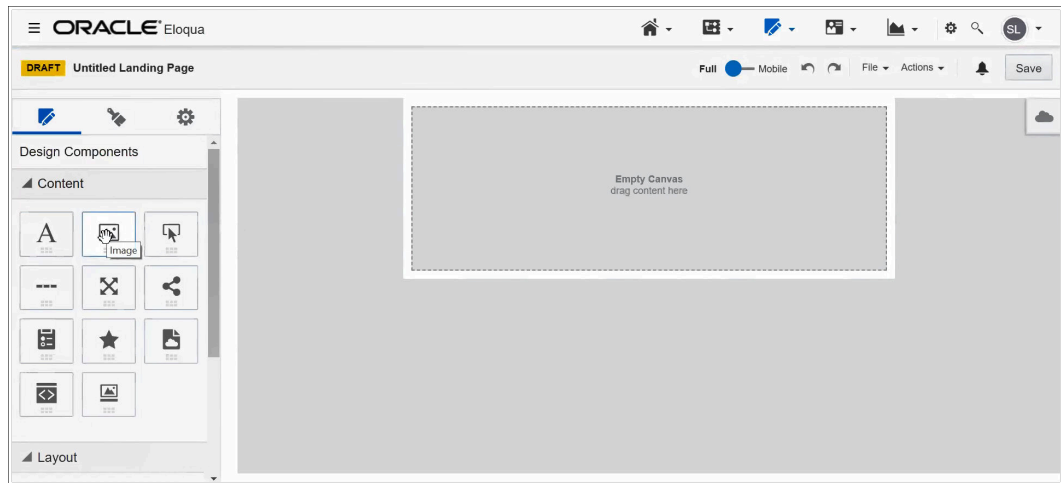
1. On the Oracle Eloqua home page, choose **Landing Pages** from a drop-down menu on the right.



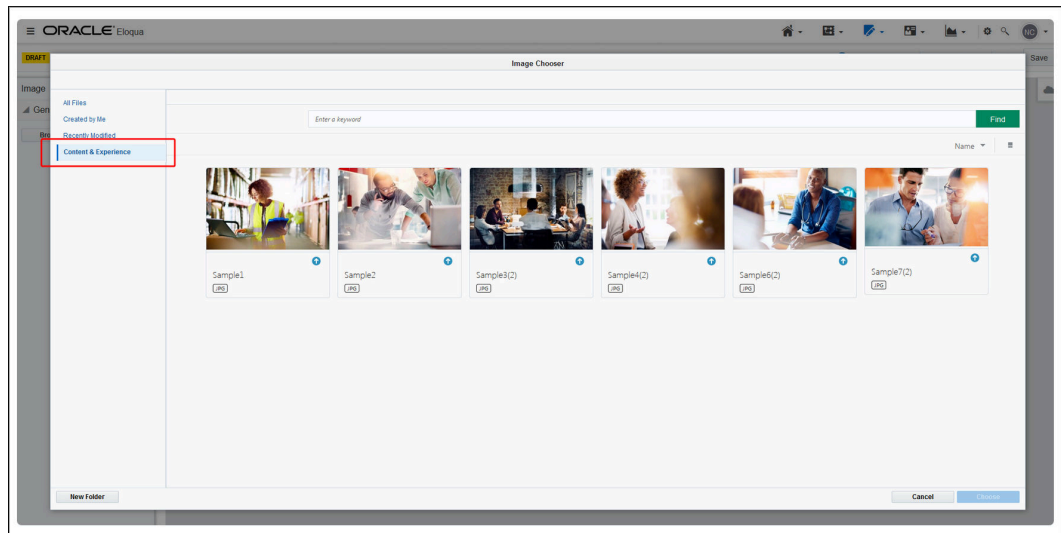
2. On the **Landing Pages** page, click **Create a Landing Page**. Eloqua displays the **Template Chooser** for a landing page.



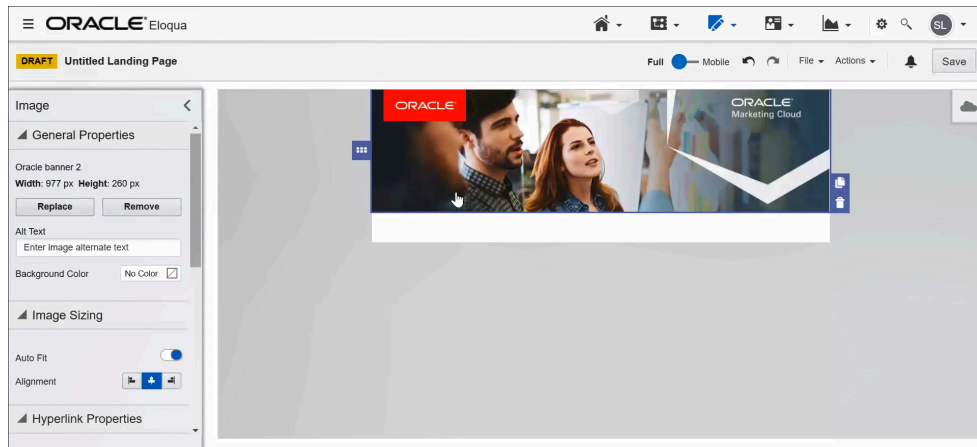
3. Click **Blank Responsive Landing Page**. This opens the editor page with the empty canvas.



4. Drag and drop the image symbol from under **Content** on the left onto the blank canvas. You can upload the image or browse for it.
5. In the Image Chooser, you can browse for an image in the Oracle Content Management repository that was created to use with Eloqua. You can select any image in the repository.



6. After you choose an image from the Oracle Content Management landing page, you can set properties for and size the image on your Eloqua landing page.



 **Note:**

Using Internet Explorer to insert assets from Oracle Content Management might cause an error. Oracle recommends that you use Google Chrome or Mozilla Firefox to insert assets from Oracle Content Management until this issue is resolved in a future release.

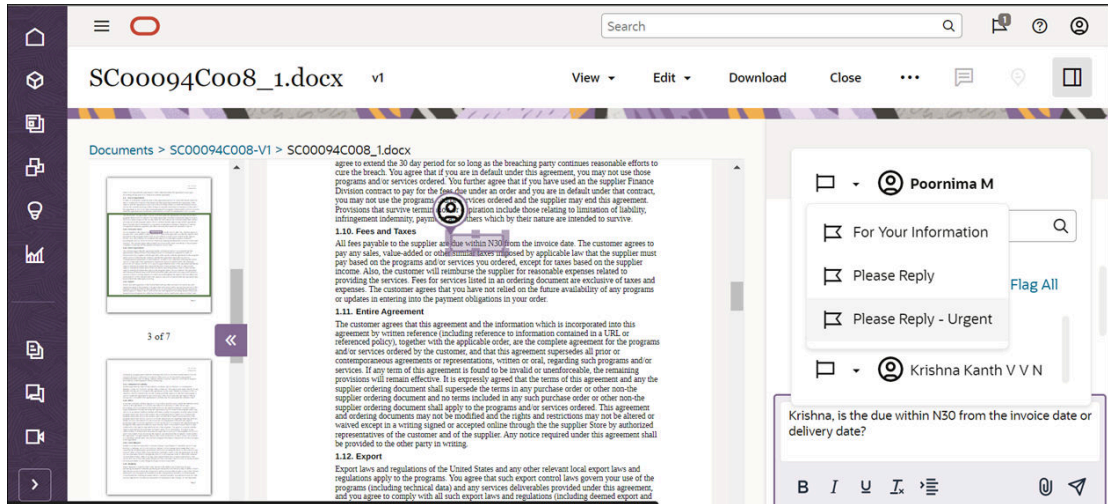
Integrate with Oracle Enterprise Contracts

Why Integrate with Oracle Enterprise Contracts?

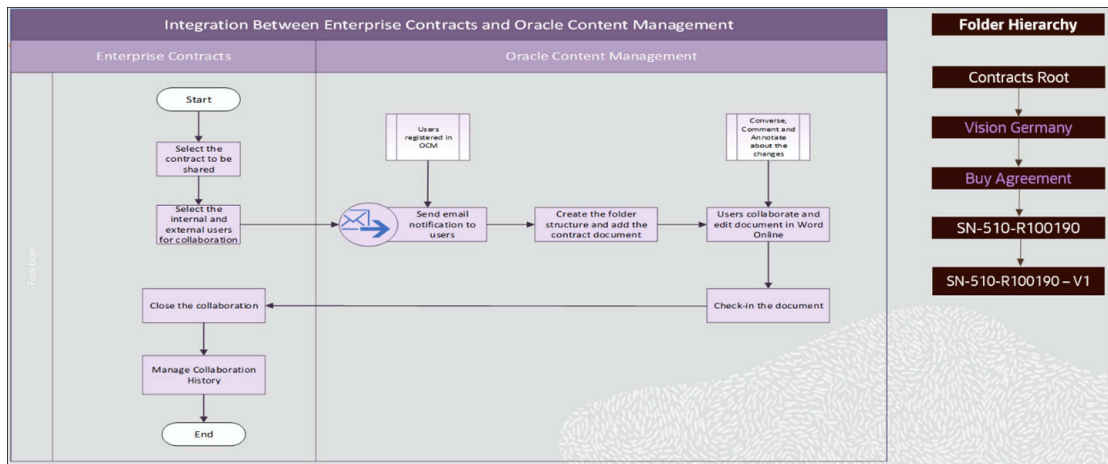
Oracle Enterprise Contracts is a comprehensive offering that standardizes corporate contract policies, improves internal controls, and enforces compliance with all contractual obligations and regulatory requirements. It provides a complete solution for managing sales, procurement, and other contracts.

The new addition to Oracle Enterprise Contracts, [Collaborate on Contract Negotiations](#), integrates Oracle Content Management (OCM) functionalities, so you (e.g., internal and external stakeholders) can take advantages of the features in both applications to [smooth the contract negotiation process](#). For instance, you can share contracts for review and revision, collaborate and edit contracts in Microsoft Word from the OCM UI. Additionally, leveraging OCM as a cloud-based content management system streamlines content creation tasks, accelerates team collaboration, promotes content reuse from a single source (the OCM [repository](#)), and improves customer and employee engagement.

The screenshot below shows the stakeholder commenting on a document in the Oracle Content Management UI.



The screenshot below shows the integration diagram between Oracle Enterprise Contracts and Oracle Content Management.



Prerequisites

There are prerequisites to integrating Oracle Content Management with Oracle Enterprise Contracts. You need the following before the integration process:

- An Oracle Cloud Account.
- A subscription that covers the use of Oracle Enterprise Contracts. The subscription depends on the types of contracts that you want to manage. Contact your [Oracle sales team](#) for assistance.
- An Oracle Content Management instance with the application roles assigned: Raise a Service Request (SR) with the Oracle Fusion Applications team in [Oracle Support](#) to configure Oracle Content Management with [Oracle Fusion](#).

 **Note:**

The Oracle Content Management instance needs to be provisioned in the same environment as your [Oracle Fusion applications](#). You cannot use an existing Oracle Content Management instance that has not been instantiated in your Oracle Fusion environment.

Integration Process

1. Enable the integration.

Once your Oracle Content Management is provisioned in the Oracle Fusion Applications environment, enable the option by using the [Opt-In UI process](#). The [Opt-In UI](#) offering is **Enterprise Contracts**.

2. Configure the root folder path and root user.

Use the task **Manage Content Management Configuration** from the contracts landing page to set up the OCM root folder path and root user before sharing contract documents for collaboration.

For complete details, see [Collaborate on Contract Negotiations using Oracle Content Management, Steps to Enable](#) in Oracle Sales Force Automation—Enterprise Contracts.

Integrate with Oracle Integration

Why Integrate with Oracle Integration—Oracle Process Cloud?

[Oracle Integration—Oracle Process Cloud Service](#) automates and manage your business work flows. Through integration with Oracle Content Management, your users can [access Oracle Integration](#) functionality within Oracle Content Management, letting users manage business processes in the cloud, such as content workflows to route content for approval or review.

 **Note:**

Oracle Content Management Starter Edition has a limited feature set. To take advantage of the full feature set, upgrade to the Premium edition.

Integrating Oracle Content Management with Oracle Integration benefits document-intensive processes by organizing, managing, and restricting access to documents that must be submitted, reviewed, and approved or rejected by different roles and organizations during the business process. Conversations enable users to easily discuss things that come up during the process.

Oracle Content Management integrates assets, content workflows, documents, and conversations with your process applications.

- **Assets:** You can enable structured multistep workflows for review and approval of content items and digital assets that you manage in Oracle Content Management asset repositories.

- **Content workflows:** Oracle Content Management includes a workflow management system that supports business process-based integration. This enables modeling, automation, and continuous improvement of business processes and routing information according to user-defined business rules.
- **Documents:** Oracle Integration provides simple file-attachment functionality, but if you need something more robust to handle document-intensive processes, you can integrate Oracle Content Management. This service enables you to organize files into folders, manage access to each folder, and even start a process when you upload a document. For example, if you're processing a home loan, you need to manage documents such as loan applications, employment histories, and house appraisals, making sure that the right users see the documents they need to submit, review, or approve, but they don't get access to restricted information.
- **Conversations:** When you integrate conversations, users can easily discuss things that come up during the process. This provides a record of what happened, enabling you to quickly bring new stakeholders up to speed or refer back to things as necessary. Plus, the conversation tools work like the social media tools users regularly use, but with enterprise-wide security and controls. For example, if you're working on a contract, you might need to discuss some of the terms while still making sure your discussion is confidential.
- **Document- and Folder-Initiated Processes:** You can automatically start a process when someone uploads a document (or folder of documents) to a chosen document folder.

Prerequisites

There are prerequisites to integrating Oracle Content Management with Oracle Integration—Oracle Process Cloud. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM.

- Check the [Oracle Integration—Oracle Process Cloud](#) documentation for any specific requirement.
- On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Integration Process

Before users can take advantage of the integrated functionality, a service administrator must configure settings in both Oracle Integration (described in [Integrate Documents](#) in *Using Processes in Oracle Integration 3*) and Oracle Content Management (described in [Configure Oracle Integration Settings in Oracle Content Management](#)).



Note:

The integration of these services requires single sign-ons (SSO), so both services must be in the same identity domain and tenancy. For full details on the concepts of identity domain and tenancy, see topics such as [Account and Access Concepts](#) and more in the Oracle Cloud Infrastructure documentation.

After both services have been configured:

- Oracle Integration users can take actions (such as approvals) on the files directly in Oracle Integration.
- Oracle Content Management content administrators can use the Oracle Content Management web interface to manage the workflows created in Oracle Integration, registering them, assigning them to repositories, adding members, and assigning workflow roles.
- Oracle Content Management content contributors can use workflows to get approval for their content.
- Oracle Content Management documents users can upload files into folders to initiate a workflow associated with the folder.
- Oracle Content Management site designers can create web pages with ready-to-use components that provide folder and file access, process selection and initiation, associated conversation display and interaction, and much more.

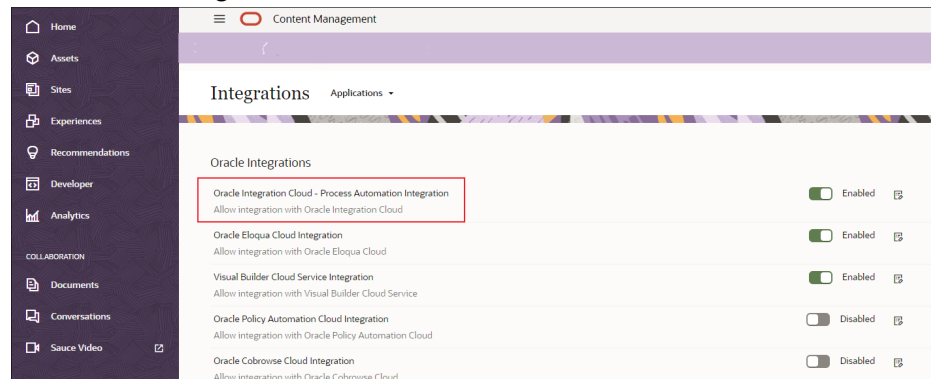
The following sections provide more details about integrating with Oracle Integration:

- [Configure Oracle Integration Settings in Oracle Content Management](#)
- [Oracle Integration with Assets](#)
- [Oracle Integration with Documents](#)
- [Oracle Integration with Sites](#)
- [Pass a CSS Style Sheet to Oracle Integration](#)
- [Start the Default Version of an Oracle Integration Process](#)

Configure Oracle Integration Settings in Oracle Content Management

To configure Oracle Content Management to integrate with Oracle Integration and to enable content workflows:

1. Enable the integration and enter connection information.
 - a. Sign in to the Oracle Content Management web interface as a service administrator.
 - b. In the **Settings** menu, click **Integrations**.
 - c. Under **Oracle Integrations**, select **Oracle Integration Cloud - Process Automation Integration** to enable the service.



- d. In the OIC Process Configuration dialog, enter the following information.

- **Service URL:** The URL of the service that users can access for their applications.
 - If you have a Universal Credits subscription, your Service URL should look something like this:

```
https://{servicename}/ic/api/process/v1/processes
```

- If you have a non-metered subscription, your Service URL should look something like this:

```
https://{servicename}/bpm/api/4.0/processes
```

 **Note:**

For asset workflows, only `https://{servicename}/bpm/api/4.0/processes` is valid. If you're still using `https://{servicename}/bpm/api/3.0/processes`, the recommendation is to use 4.0 version of the service URL.

- **Service User:** Enter the email address of the user who owns the process to be used in Oracle Content Management. This must be the same user you entered when configuring Oracle Integration.
- **Service Password:** Enter the user password. This must be the same password you entered when configuring Oracle Integration.
- **Client ID:** Enter the Client ID of the App from the (Identity Cloud Services) IDCS Admin Console corresponding to the Oracle Integration instance.

 **Note:**

To find the Client ID, follow the steps below. The UI of the IDCS Admin Console can change, treat these steps as guidance.

- i. Log in to IDCS Admin Console.
 - ii. Click on **Oracle Cloud Services**.
 - iii. Locate application corresponding to your Oracle Integration Cloud.
 - iv. Go to **Configuration** and in the **General Information** section you will find Client ID.
- e. Select the **Enable Workflows for Assets** checkbox.
2. In Oracle Content Management, enable Oracle Integration use for the desired folders.
- a. In Oracle Content Management, open the properties for the folder.
 - b. Enable Oracle Integration use.
 - c. Select a process from the list.

If the process list is blank, it's caused by one of the following issues:

- The Oracle Integration user you specified doesn't have rights to see the processes.
- The Oracle Integration URL you specified isn't correct.
- The Oracle Integration user/password combination you specified isn't correct.
- The Oracle Integration service doesn't have a process that uses a Documents Start Event. To create a process with a Document Start Event, see [Creating a Document- or Folder-Initiated Process](#) in *Using Processes in Oracle Integration 3*.

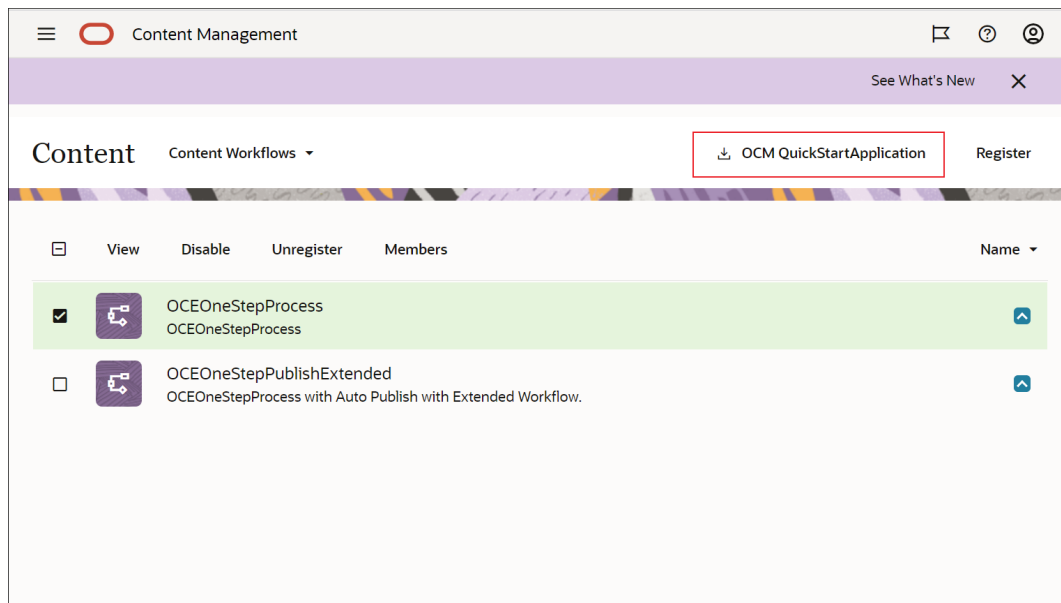
Oracle Integration with Assets

After [integrating Oracle Integration – Process Automation with Oracle Content Management](#), you can create structured multistep workflows for review and approval of content items and digital assets that you manage in Oracle Content Management asset repositories.

Oracle Content Management provides a quick start application package with several multistep processes that you can deploy to your Oracle Integration instance and start using for asset review and approvals. Alternatively, you can use these processes as samples to develop custom processes that meet specific requirements in your organization. For example use cases, see *Use Seeded Content Workflows in Managing Assets with Oracle Content Management*.

To download the quick start application package and set up the processes for use:

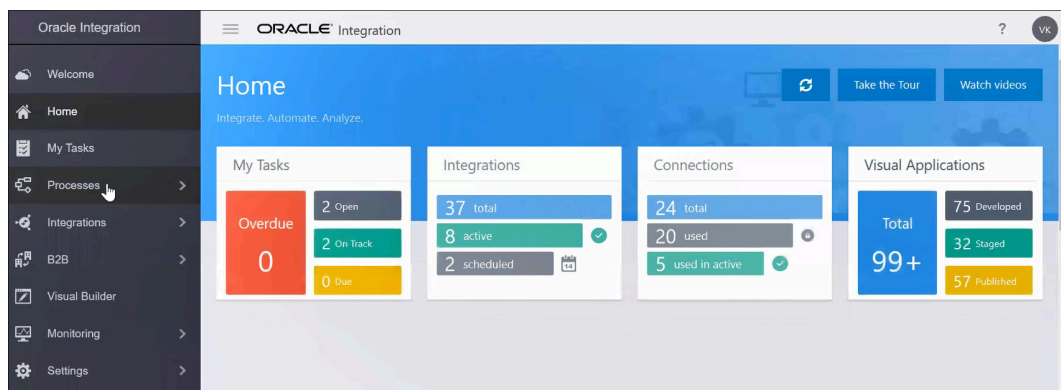
1. In the Oracle Content Management web user interface, on the **Content > Content Workflows** page, click the **OCE QuickStartApplication** link to download a zip file.



2. Unzip the file to a local computer.

After downloading and unzipping the QuickStartApplication package, you must go to Oracle Integration to import an application for use with Oracle Content Management.

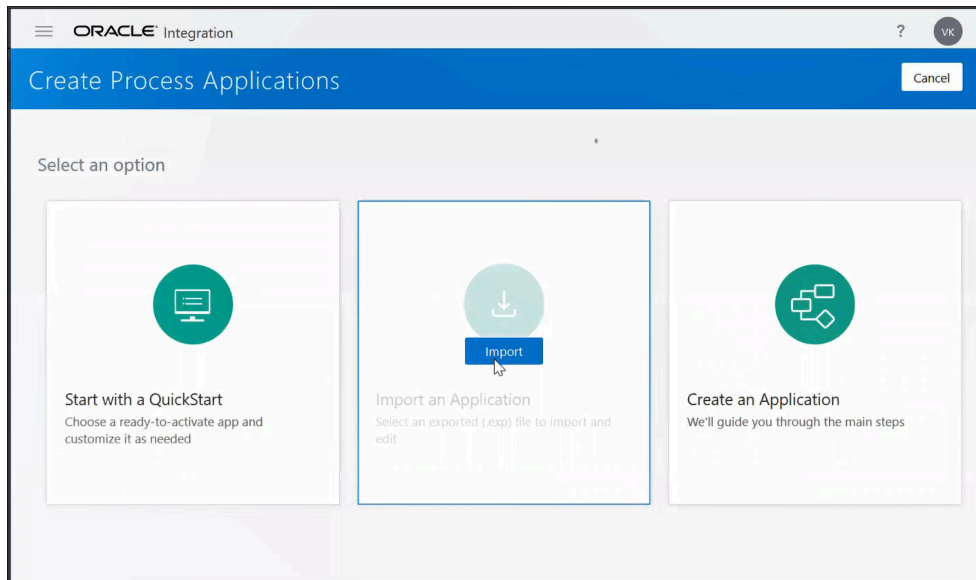
1. Sign in to Oracle Integration (OIC).



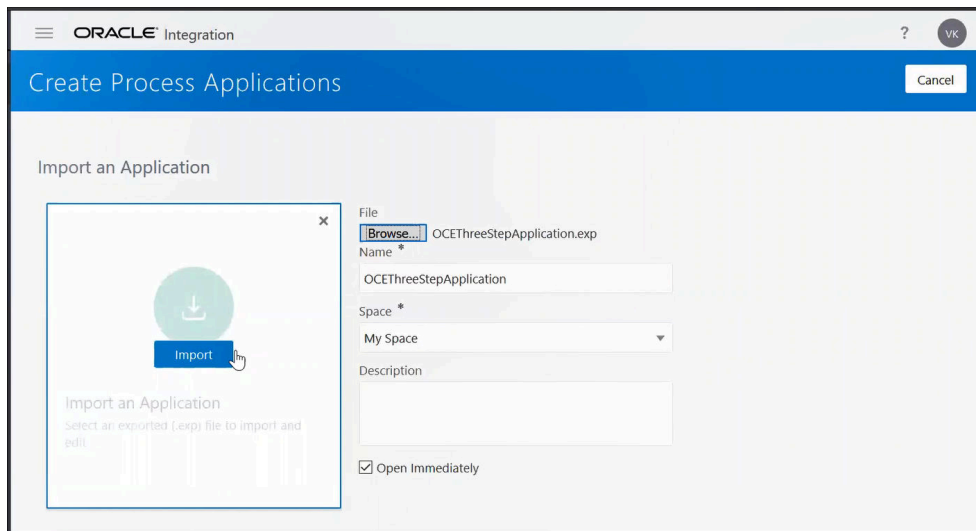
2. Click **Processes > Process Applications**.

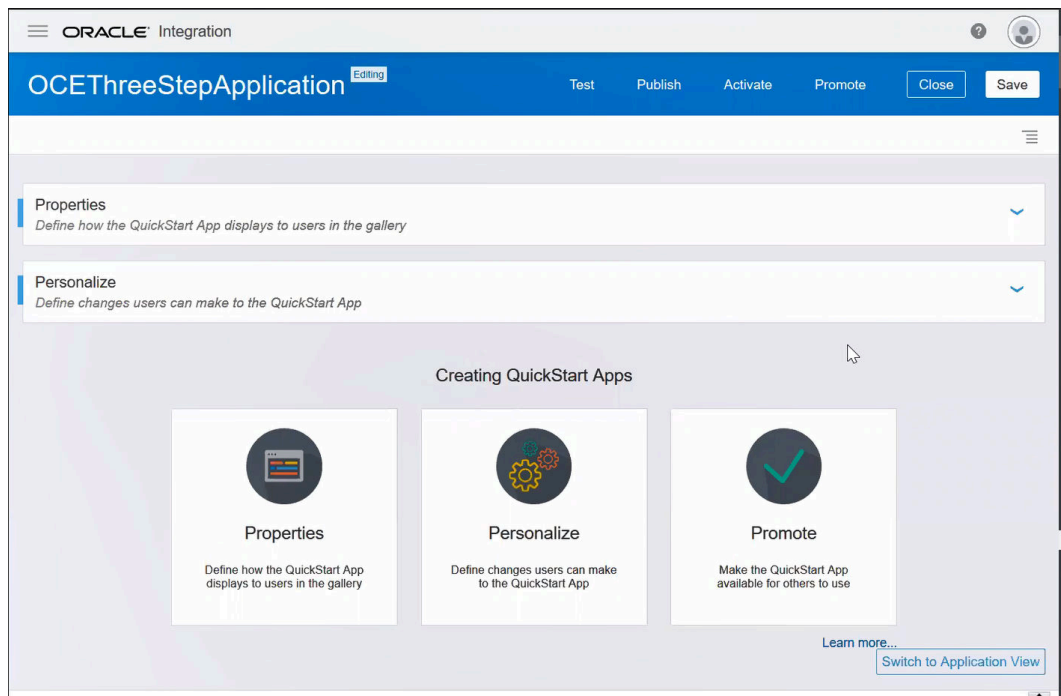
3. Click **Create**.

4. On the **Import an Application** tile, click **Import**.



5. Click **Browse**.
6. Point to one of the three items that were unzipped from the zip file you downloaded earlier; for example, **OCEThreeStepApplication.exp**.
7. Click **Import**.





The application is imported as **QuickStart Master**, You have the option to change the name before you promote it to be a QuickStart application:

1. Without any changes, click **Promote**.
2. In the **Promote to Gallery as a Quick Start App** dialog, enter a snapshot name and then click **Promote**.
3. After the process completes, click **Close** in the dialog.
4. Click **Close** in the main window.

Now you can create another QuickStart application and use that application to create multiple process applications:

1. On another **Process Applications** page, click **Create**.
2. On the **Start with a QuickStart** tile, click **Browse**.
In the Gallery, you should now see the application sample you just created (for example, **OCETHreeStepApplication**).
3. Click **Create** for that application. Enter a name, and then click **Create**.
An overview page for the new application displays.
4. Click **Configure**.
5. Click **Update your OCE Connection URL here. This is mandatory**.
6. In the **Address** field, replace **OCEURL** with your Oracle Content Management instance URL (copy and paste from your instance in a web browser).
So you'll end up with something like `https://instance.url.oraclecloud.example.com/content/management/api/v1.1`.
7. Click **X** to close the page.

There is no need to modify anything else at this point. You can click **Switch to Application View** and then click the name of an application (for example, **OCETHreeStepProcess**). The application process opens in a process flow visualizer.

You could modify the roles here. Make sure to click **Save** if you make any changes.

To deploy an application, click the house icon in the top left to go to the OIC home page, and then follow these steps:

1. Click **My Tasks** in the left navigation pane.
2. Click **Administration** in the left navigation pane.
3. Click **Manage Credentials**.
4. Click **Add new credential**. Enter the user name and password for your Oracle Content Management repository administrator. You'll use this keystore when you activate the application.
5. Go back to the home page.
6. Click **Processes > Process Applications**.
7. Click the name of a QuickStart application.
8. On the application overview page, click **Publish**.
9. In the **Publish Application** dialog, enter a comment, if necessary, and then click **Publish**.

To activate the application, follow these steps:

1. Click **Switch to Application View**.
2. Click **Activate**.
3. On the **Activation** tab, click **Activate new version**.
4. In the **Activate Application to My Server** dialog, click **Customize**.
5. Select the keystore credential you created for the Oracle Content Management repository administrator.
This will populate the **user** and **password** fields.
6. Click **Validate**.
7. If there are no errors, click **Options**.
8. Add the version (for example, **1.0**) and then click **Activate**.
This will deploy the process application in Oracle Integration and make it available in Oracle Content Management.
9. Click **Finish** after the application is activated successfully.

After the application is registered, have your Oracle Content Management content administrator use the Oracle Content Management web interface to make the workflow available for use, registering it, assigning it to a repository, adding members, and assigning workflow roles. Then Oracle Content Management content contributors can use the workflow to get approval for their content.

Oracle Integration with Documents

Manage workflows for business applications, such as document routing for review or approval, with Oracle Integration enabled for documents in Oracle Content Management.

You can allow your users to access Oracle Integration functionality, which lets users manage business processes in the cloud, such as document routing for approval or review.

This feature might not be available, depending on the Oracle Content Management subscription type and start date of your service.

You must configure settings in both Oracle Integration and Oracle Content Management before users can take advantage of the integrated functionality.

1. In Oracle Integration, sign in as an administrator and enter connection information for Oracle Content Management. See [Integrating Documents in Using Oracle Process Cloud Service](#).
2. In Oracle Content Management, enable Oracle Integration and enter connection information:
 - a. Sign in to Oracle Content Management as an administrator.
 - b. From the **Administration** menu, choose **Integrations**.
 - c. Under **Applications**, select **Oracle Integration** to enable the service, and then set these values:

- **Service URL:** The URL of the service that users can access for their applications.
 - If you have a Universal Credits subscription, your Service URL should look something like this:

```
https://{servicename}/ic/api/process/v1/processes
```

- If you have a non-metered subscription, your Service URL should look something like this:

```
https://{servicename}/bpm/api/4.0/processes
```

 **Note:**

For asset workflows, only `https://{servicename}/bpm/api/4.0/processes` is valid. If you're still using `https://{servicename}/bpm/api/3.0/processes`, the recommendation is to use 4.0 version of the service URL.

- **Service User:** Enter the email address of the user who owns the process to be used in Oracle Content Management. This must be the same user you entered when configuring Oracle Integration.
- **Service Password:** Enter the user password. This must be the same password you entered when configuring Oracle Integration.
- **Client ID:** Enter the Client ID of the App from the (Identity Cloud Services) IDCS Admin Console corresponding to the Oracle Integration instance.

 **Note:**

To find the Client ID, follow the steps below. The UI of the IDCS Admin Console can change, treat these steps as guidance.

- i. Log in to IDCS Admin Console.
- ii. Click on **Oracle Cloud Services**.
- iii. Locate application corresponding to your Oracle Integration Cloud.
- iv. Go to **Configuration** and in the **General Information** section you will find Client ID.

3. To create a process with a Document Start event, see [Creating a Document- or Folder-Initiated Process](#) in *Using Oracle Process Cloud Service*.

When a task step is complete, the file can be managed according to the defined process. For an incoming document, a user can perform actions based on the assigned role for that document: Contributor, Downloader, or Viewer.

When Oracle Content Management starts a process, the following payload is sent to launch the process:

```
{
  "operation": "startEvent",
  "processDefId": "testing~LoanApplicationProcessing!
1.0~LoanApplicationProcessing"
  "params": {
    "id": "abc123",
    "name": "document name",
    "startedBy": "user id",
    "type": "d",
    "role": "role that should be used to generate
subsequent applinks",
    "version": "version"
  }
}
```

As a developer, you need to be aware of the following requirements for the process you develop:

- It needs to be a process that uses an Oracle Content Management Document Start event.

- When deploying the process, you need to share it with the user specified for enabling the integration so that user has the rights to trigger the process,
- For the user who uploaded the file to show up as the user who started the task, the process must use the value passed in the **startedby** field as the display name for the initiator.

Oracle Integration with Sites

You can integrate tasks on your sites with Oracle Integration.

Developers need to be aware of the following requirements for Oracle Integration with Sites:

- The process author needs to ensure that the sites author is added as an initiator of the process, or the site author will not be able to see the process in the list of processes available when configuring the start form component.
- The process author also needs to ensure that any site visitor is added as an initiator of the process, or the visitor, upon filling in the start form, will not actually be able to initiate the process.

Pass a CSS Style Sheet to Oracle Integration

As a developer, you can control the look of an Oracle Integration start form in a site by passing CSS information through the `design.css` file in a theme.

To do this, you need to place the CSS style inside of the `design.css` file that is within the theme of the site. The style sheet would most likely be related to the theme.

On a site page, you can drop an Oracle Integration form with **Name** and **Address** fields. There is no styling for the fields.

1. Click **Edit** in the top right menu to go into edit mode.
2. On the **Process Start Form** menu, choose **Settings**.
3. In the **Process Start Form Settings** dialog, click **Custom Settings**.
4. In the **Properties** panel, you can choose a control class under **Control Class Name** for each field in the Oracle Integration form. Each control class specifies a CSS style.
5. In the `design.css` file for the theme of a site, you can specify properties for each control class, such as a bold label.
When developing a form, you can specify the control class name on a field-by-field basis.

On the **Style** tab in the **Button Settings** panel, contributors can choose styles that come with a theme. You can set options for a custom style in Site Builder for editing a site.

In the text editor toolbar for both paragraphs and titles, you can choose styles for toolbar groups. The specifications for these style options go in the `components.json` file for the theme.

Start the Default Version of an Oracle Integration Process

You can start the default version of an Oracle Integration process and view a sorted list of Oracle Integration start forms to choose from.

The Oracle Integration start form is part of a process, and the process is an application. When you activate the application package, it has an associated version number.

1. Each time you change and publish an Oracle Integration process, click **Activate** in the top menu to activate the latest version of the process, which becomes the default version.
2. Under Select a Process in the Custom Settings panel, check **Use default process version**.
3. Select a process, and choose a start form from the list.



Note:

Only one experience connection can be created on an Oracle Content Management Starter Edition instance. For unlimited connections, upgrade to Premium Edition.

Integrate with Oracle Intelligent Advisor

Why Integrate with Oracle Intelligent Advisor?

The integration of Intelligent Advisor with Oracle Content Management (OCM) lets you add an Intelligent Advisor component to a site page in OCM. As a result, you give users access to [Oracle Intelligent Advisor](#) (formerly Oracle Policy Automation) functionality, which implements online "interview" scenarios, such as feedback for troubleshooting or eligibility assessments for services. Intelligent Advisor delivers advice across channels by capturing rules in Microsoft Word and Excel documents, then building interactive customer service experiences called interviews around those rules.



Note:

Only administrators with the *enterprise user* role can enable integration with Intelligent Advisor. If you aren't an enterprise user, the **Oracle Policy Automation Cloud Integration** option is grayed out in Oracle Content Management.

Prerequisites

There are prerequisites before you begin the integration process. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. Please also see the [Intelligent Advisor documentation](#). On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Integration Process

On the Intelligent Advisor side, interviews must be created and stored on the host site. In addition, the Intelligent Advisor administrator must add the Oracle Content Management domains (*.documents.* and *.sites.*) to the list of hosts authorized to embed interviews. See Configure security for embedded interviews in the [Intelligent Advisor documentation](#).

On the Oracle Content Management side, you need to configure integration with Intelligent Advisor:

1. After you sign in to the Oracle Content Management web application as an administrator, open your user menu and click **Administration**.
2. In the **Administration** menu, click **Integrations**.
3. Under **Oracle Integrations**, select **Oracle Policy Automation Cloud Integration** to enable the service, and then set these values:
 - **Service URL:** Enter the URL of the Intelligent Advisor Cloud Service.
 - **Service User:** Enter the name of the Intelligent Advisor user. This user must be an Integration user and must have the *Deploy Admin* role for the Intelligent Advisor collections. See [Create an account for application integration in the Intelligent Advisor documentation](#).
 - **Service Password:** Enter the user password.

After both services have been configured for integration, Oracle Content Management users can add an Intelligent Advisor component to site pages.

Integrate with Oracle JD Edwards

Why Integrate with Oracle JD Edwards?

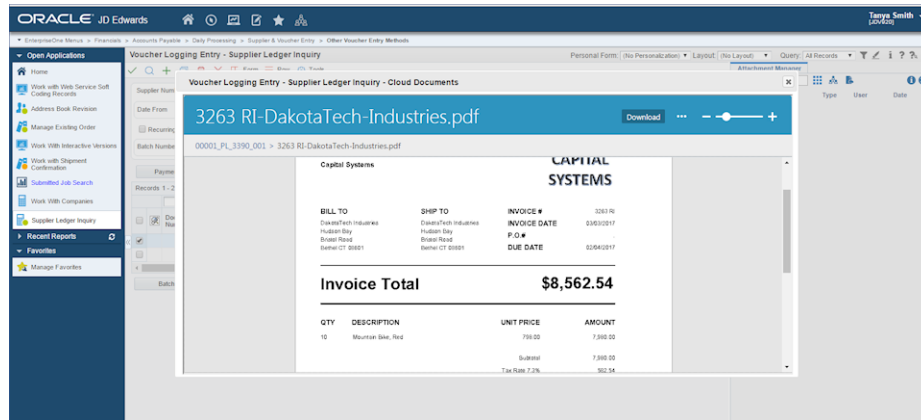
The integration of [Oracle JD Edwards EnterpriseOne](#) with Oracle Content Management lets you attach managed documents to transactions and collaborate through conversations.

With content stored in Oracle Content Management, you can do the following:

- Access documents using web, desktop devices, and mobile devices.
- View, search, and manage documents directly in the web interface.
- Collaborate through conversations, and conversations about specific transactions or documents.
- Review document usage data.

When you enable the integration, you can use the following from within JD Edwards EnterpriseOne:

- **User Conversations:** Start or participate in social conversations about a topic of interest within EnterpriseOne using the Conversation icon in the menu bar.
- **Contextual Conversations:** Start or participate in conversations about specific business records within EnterpriseOne. Create conversations within the context of a transaction using the Conversation icon in the header bar of the Attachment Manager.
- **Contextual Documents:** Add documents to the cloud and organize documents while still keeping them in the context of your business record by using the Document icon in the header bar of the Attachment Manager.



Prerequisites

To integrate Oracle Content Management with JD Edwards, there are [prerequisites](#) from the JD Edwards side. On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Integration Process

Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. For more integration details, please also refer to the [Oracle JD Edwards documentation](#).

Integrate with Oracle Logistics Cloud

Why Integrate with Oracle Logistics Cloud?

You can use Oracle Content Management to store and manage documents from within [Oracle Logistics Cloud](#). The following example shows one of the Oracle Logistics Cloud solutions, Oracle Transportation and Global Trade Management, being integrated with Oracle Content Management.

Prerequisites

These are the prerequisites before you begin the integration process.



Note:

Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. Please also see the [Transportation and Global Trade Management 22C](#) documentation.

- Create and activate an Oracle Cloud Account before you begin.
- Review Overview of Oracle Content Management, which provides information on interacting with OCM and concepts on access roles for performing certain tasks.

- Compare the starter vs. the premium edition of Oracle Content Management. Only the premium edition has all the features.
- Create an Oracle Content Management instance.
- Take a guided tour of [repositories in Oracle Content Management](#).
- Create an asset repository. To set this up, you need to have the repository administrator (CECRepositoryAdministrator) role in Oracle Content Management.

Integration Process

You simply configure the integration with URL, user name, and password to connect to the Oracle Content Management server. After authentication credentials are verified, you can upload, edit, and delete documents directly from Oracle Logistics Cloud. See Transportation and Global Trade Management documentation for more instructions on How to Configure Storage and How to Configure Content Organization.

Shipment Document			
Document BILL OF LADING		Review Status IN REVIEW	Instructions OPTIONAL
Revisions			
	Revision	Review	Annotation
	SHP-TEST-001-BOL-01	DENIED	Missing proper handling instructions
	SHP-TEST-001-BOL-02	PENDING	
Status			
Type		Value	
GENERATE		GENERATE_GENERATED	
Text Templates			
Text Template	Value	For Ship Unit	For Line
HAZMAT	A test of hazardous material		
TEST	A test of ship unit line text	SU-001	1
Reviewers			
Reviewer		As Involved Party	
ESL		LOGISTICS	
SHB			
Subscribers			
Subscriber		As Involved Party	
ESL		LOGISTICS	

Integrate with Oracle Maxymiser

Why Integrate with Oracle Maxymiser?

The integration of [Oracle Maxymiser](#) with Oracle Content Management (OCM) enables you to insert assets from OCM into Oracle Maxymiser campaigns. This allows you to leverage Oracle Content Management's extensive asset repository capabilities to store content, while using Maxymiser to design campaigns.

 **Note:**

You may see Oracle Content Management being referred to as Oracle CX Content in the Maxymiser documentation.

 **Video**

Prerequisites

There are prerequisites to integrating Oracle Content Management with Oracle Maxymiser. On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Integration Process

Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. For more integration details, please also refer to the [Oracle Maxymiser documentation](#).

Integrate with Oracle Responsys

Why Integrate with Oracle Responsys?

The [integration with Oracle Responsys](#) lets you insert published assets from an Oracle Content Management asset repository into Responsys Email and Mobile campaigns. Using Oracle Content Management as a single source for all images saves rework because you can find your images easily.

Responsys users can leverage the extensive asset repositories in Oracle Content Management to store content while using Responsys to design campaigns. When you design Email and Mobile campaigns, this integration gives you the option to insert published image assets from Oracle Content Management into your campaigns. You can choose digital assets from an Oracle Content Management publishing channel through the Responsys application.

 **Video**

Prerequisites

These are the prerequisites before you begin the integration process.

 **Note:**

Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. Please also see the [Oracle Responsys documentation](#).

- Create and activate an Oracle Cloud Account before you begin.

- Review Overview of Oracle Content Management, which provides information on interacting with OCM and concepts on access roles for performing certain tasks.
- Compare the starter vs. the premium edition of Oracle Content Management. Only the premium edition has all the features.
- Create an Oracle Content Management instance.
- Take a guided tour of [repositories in Oracle Content Management](#).
- Create an asset repository. To set this up, you need to have the repository administrator (CECRepositoryAdministrator) role in Oracle Content Management.

Integration Process

The process of Responsys integrating with Oracle Content Management consists of the following steps:

1. [Choose an Asset Repository and Create Two Publishing Channels](#)
2. [Enable the Integration](#)

Then you can create and publish assets in Oracle Content Management and view images in the Responsys Message preview.

Choose an Asset Repository and Create Two Publishing Channels

When you design an Email or Mobile campaign, you will choose assets from the publishing channel for the appropriate campaign type. Responsys will automatically filter content assets in Oracle Content Management according to the campaign you are designing.

After you choose an asset repository in Oracle Content Management, create and share two publishing channels: a mobile channel and an email channel. Oracle suggests creating channels that are specifically for Responsys. See [Create and Share Publishing Channels](#).

In Responsys, you can choose Oracle Content Management assets from these channels.

Enable the Integration

You can enable the integration between Oracle Responsys and the Oracle Content Management account from Responsys.

In Oracle Responsys, choose **Oracle Content Management Cloud Integration** from the **Integrate** menu.

This feature is available only if it is enabled for your account.

Configure an Oracle Content Management Cloud account

1. From the side navigation bar in Responsys, select **Account**.
2. Select **Integration settings**, and then choose **Content Management Cloud settings**.
3. Provide the following details about your Oracle Content Management instance to configure your publishing channels:
 - **CEC API Version:** Your Oracle Content Management version.
 - **CEC Domain:** The service URL of the OCM instance, for example, `https://<Your OCM instance-tenancy>.cec.ocp.oraclecloud.com`

- **CEC Publish Server URL Prefix:** The service URL of the OCM instance, for example,
`https://<Your OCM instance-tenancy>.cec.ocp.oraclecloud.com`
- **Email Channel ID:** The ID of the Email channel, which is automatically generated when you create the channel.
- **Email Channel Token:** The Email channel token, which is automatically generated when you create the channel.
- **Mobile Channel ID:** The Mobile channel ID, which is automatically generated when you create the channel.
- **Mobile Channel Token:** The Mobile channel token, which is automatically generated when you create the channel.

4. Click **Save**.

Enable Oracle Content Management Embedded Content

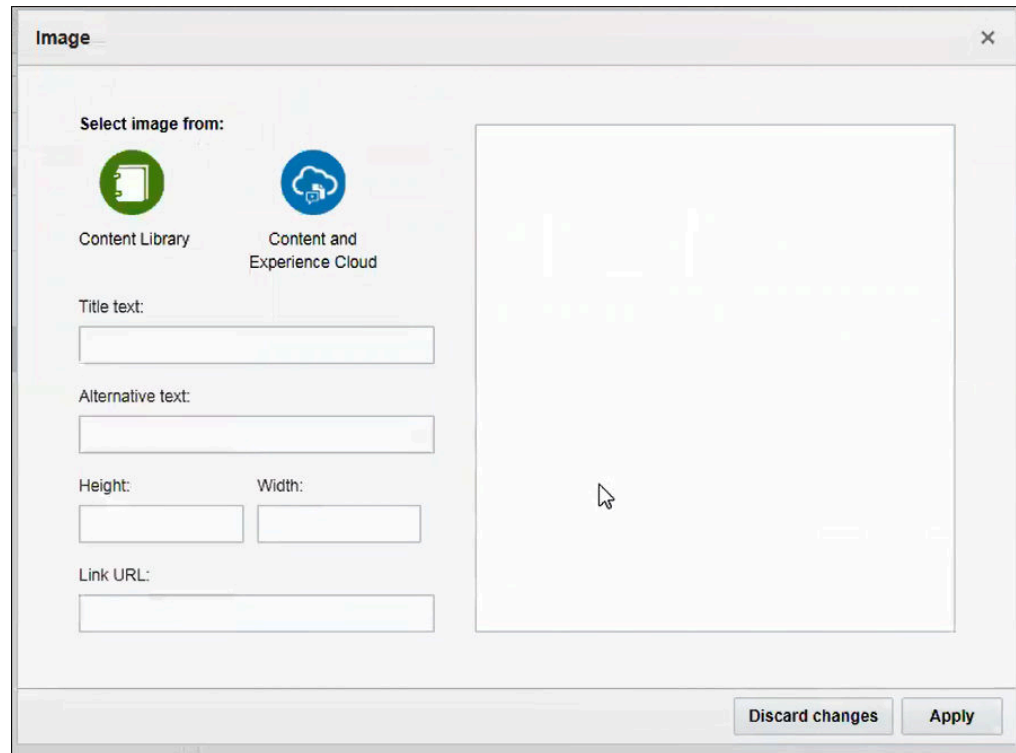
In Oracle Content Management administration security settings, enable embedded content and add the Responsys URL. Then you can embed the Oracle Content Management web UI in Responsys.

See [Embed UI API V2 for Oracle Content Management](#).

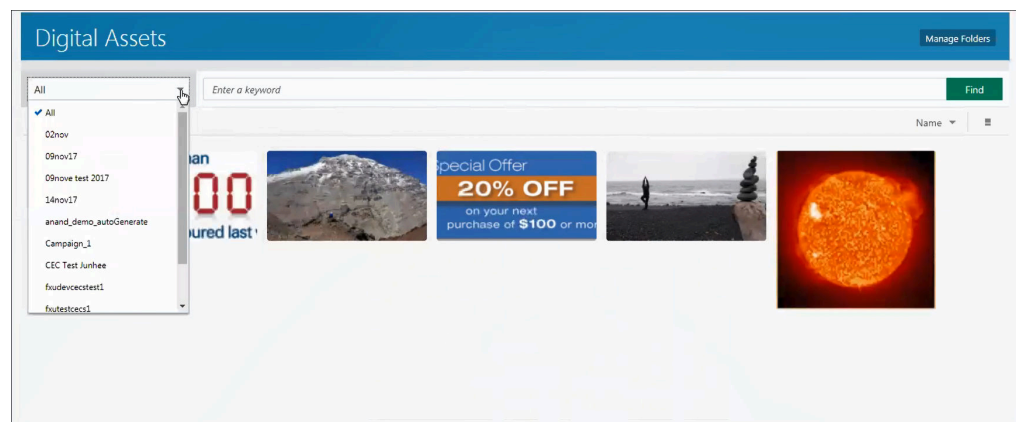
Create and Publish Assets in Oracle Content Management

You can create and publish assets in Oracle Content Management to use in Responsys.

1. Create an HTML file.
2. Edit the HTML file in your ResponSys rich text editor.
 - a. Open the image browser.
The browser displays the available image libraries:



- b. Display the Oracle Content Management asset picker.
The asset picker opens on the **Digital Assets** page, which displays published assets in the Oracle Content Management account. If you don't see an asset you want, you can use the drop-down menu on the left to navigate different asset collections. Or you can upload and publish an image in Oracle Content Management.



- c. Pick one of the assets, and click **OK**.
The **Image** dialog opens and displays a thumbnail of the image and properties that you can set for it:
- **Title text**
 - **Alternative text**
 - **Height**

- **Width**
 - **Link URL**, which makes the image clickable.
- d. Set the properties you want.
- e. Click **Apply** to insert the image into your content.
To view the URL to the image in Content Management, you can click **Source**.
In a live production environment, this would be an asset pointing directly to the content delivery network, so that the asset is accessible.

View Images in Responsys Message Preview

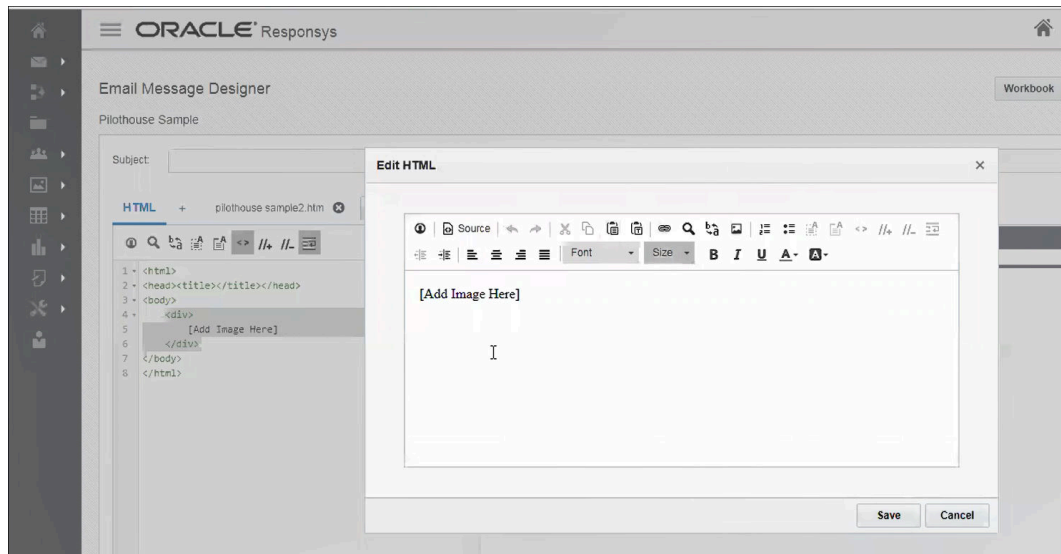
To preview Oracle Content Management images in Responsys, you can use the Responsys Message Preview.

1. On the Oracle Responsys **Manage campaigns** page, choose **Create campaign** from the drop-down menu on the right, and create an Email or Mobile campaign.

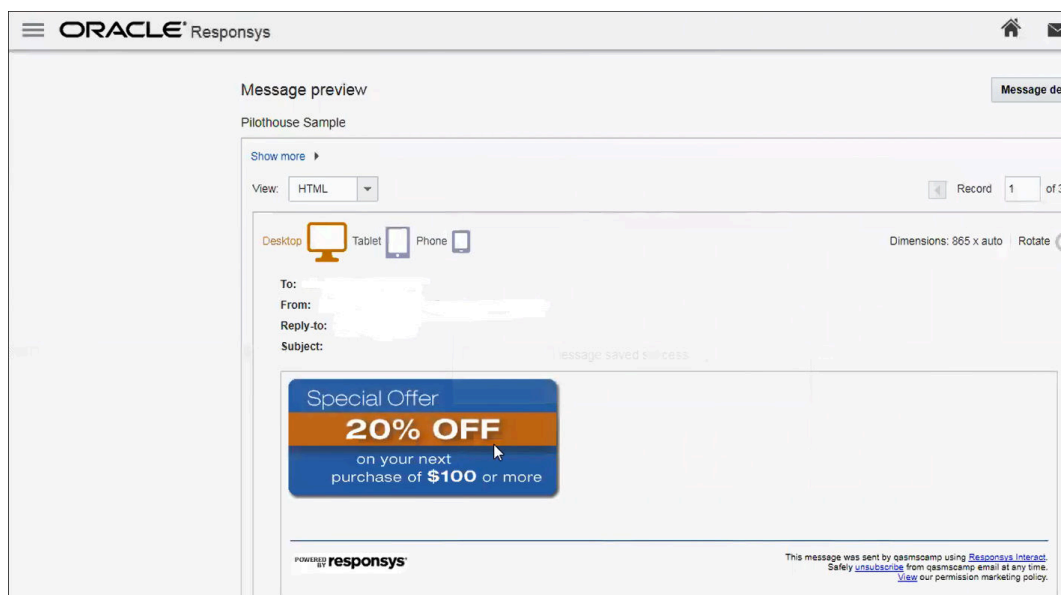
The screenshot shows a 'Create campaign' dialog box with the following elements:

- Title:** Create campaign
- Question:** What kind of campaign do you want to create?
- Options:** Five colored buttons representing campaign types: Email (orange), Push (blue), In-App (purple), SMS (pink), and MMS (green).
- Form Fields:**
 - Classic email
 - * Name:** Pilothouse Sample
 - Description:** (Empty text area)
 - * Purpose:** Promotional (dropdown menu)
 - Marketing Strategy:** --None-- (dropdown menu)
 - Marketing Program:** --None-- (dropdown menu)
 - * Folder:** Nik (dropdown menu)
 - List:** NM_Test_List (dropdown menu)
- Buttons:** Cancel and Done (with a mouse cursor over it).

2. Create a new HTML document.
3. Add a block in the HTML file for an image.



4. After the image is visible, you can click **Preview** to preview it in an email message.



Integrate with Oracle Visual Builder

Why Integrate with Oracle Visual Builder?

You can allow your users to access Oracle Visual Builder Cloud Service functionality. Oracle Visual Builder Cloud Service (VBCS) is a hosted environment for your application development infrastructure. It provides an open-source standards-based integration to develop, collaborate on, and deploy applications within Oracle Cloud. This enables users to rapidly create web and mobile applications with minimal to no coding.

 **Note:**

- Integration between these services requires SSO, so both services must be in the same identity domain.
- Only administrators with the enterprise user role can enable integration with VBCS. If you aren't an enterprise user, the **Visual Builder Cloud Service Integration** option is grayed out.

Prerequisites

There are prerequisites to integrating Oracle Content Management with Oracle Visual Builder. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM.

- Check the [Oracle Visual Builder documentation](#) for any specific requirement.
- On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.

Integration Process

On the VBCS side, the following must be done before this feature can be used with Oracle Content Management:

- Cross-Origin Resource Sharing (CORS) must be enabled on the VBCS site.
- Apps [must be created and made available](#) for embedding.
- The apps must be [configured](#) for use with Oracle Content Management.
- Web applications must be created and made available for embedding in an iframe.
- The Sites SDK must be imported.
- The Sites SDK must be referenced in the web applications.
- A page URL parameter called "id" must be added to the web applications.

On the Oracle Content Management side, you need to configure integration with VBCS:

1. After you sign in to the Oracle Content Management web application as an administrator, click **Integrations** in the Administration area of the navigation menu.
2. Under **Oracle Integrations**, select **Visual Builder Cloud Service Integration** to enable the service.
3. Enter the **Service URL** of the Oracle Visual Builder Cloud Service.

 **Note:**

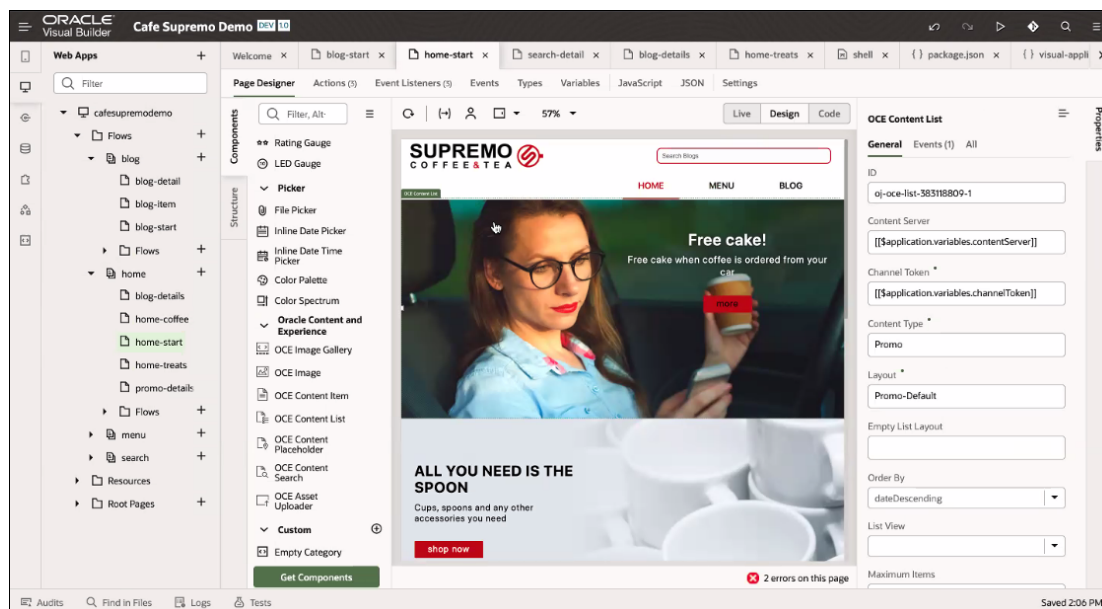
If you have Universal Credits subscription, you must include `ic/builder` in your Service URL. For example, `https://vbcserver.example.com/ic/builder`.

After both services have been configured for integration, Oracle Content Management users can create components for your VBCS apps and add them to site pages.

- [Use Oracle Content Management Components in Oracle Visual Builder Applications](#)
- [Embed Oracle Visual Builder Applications in an Oracle Content Management Site Page](#)
- [Build an Oracle Content Management VBCS Form and Data Report Components](#)
- [Build an Oracle Content Management VBCS Secure Form Component](#)
- [Provide a VBCS Endpoint As a URL for Select Menus](#)

Use Oracle Content Management Components in Oracle Visual Builder Applications

Oracle maintains a component exchange containing components validated by Oracle that are publicly available to all developers. A component exchange is a repository of custom components available in VB Studio and includes several components designed to use with Oracle Content Management. You can use these components in your visual applications.



To integrate a component exchange with a Visual Builder instance, you provide the exchange's URL and credentials in the Tenant Settings. For detailed information about integrating a Visual Builder component exchange, see [Manage Your Component Exchange](#) in the *Administering Oracle Visual Builder* guide.

Cross Origin Resource Sharing (CORS) Requirements

The Oracle Visual Builder origin hosting the Oracle Content Management components for Oracle Visual Builder must be granted explicit permission to make [cross-origin requests](#). Contact the service administrator for the Oracle Content Management instance that you want to access to request that the Oracle Visual Builder origin be added to the Front Channel CORS Origins list. For more information about cross-origin requests.

Secure Channel Requirements

Some components, to access a secure channel, also require configuration steps to be taken in both Oracle Identity Cloud Service (IDCS)/IAM Identity Domain and Oracle Visual Builder:

- Set up the configuration for authentication in Oracle Identity Cloud Service (IDCS)/IAM Identity Domain.
- Configure the service connection in Oracle Visual Builder.

This grants the authentication token used to access secure Oracle Content Management endpoints. Once this is completed, the component will be able to consume the service connections created.

Detailed information about if a component needs to access a secure channel and the configuration steps needed to do so can be found in each component Read Me file.

Connecting to Oracle Content Management

To use Oracle Content Management components for your Visual Builder application, you need to provide the Oracle Content Management URL. This can be provided by the Oracle Content Management service administrator, or you can navigate to the instance and copy the URL. Don't worry about subdirectories. For example, both of these will work:

- <https://sample.cec.ocp.oraclecloud.com/documents/home>
- <https://sample.cec.ocp.oraclecloud.com>

This property will be initialized as an application-level variable by default when you add the component to your page. You can then go to your application-level variables and change the default value to your actual value. Once the variable is updated in Oracle Visual Builder, when you add another OCE component that requires this attribute, the attribute will default to the same variable initiated by the first component that was imported.

Provide a Channel Token

OCE components that place assets on an Oracle Visual Builder application page work only for published content items. Content items must be published through a publishing channel. When you publish a content item, you'll be forced to choose a publishing channel. To get the channel token, you can do the following:

1. Log in to the Oracle Content Management web interface as an administrator.
2. Click **Content** in the left navigation panel (under "Administration").
3. Choose **Publishing Channels** from the dropdown list.
4. Select the publishing channel you'll be using, or create a new one.
5. Choose **Edit** from the list of available actions.
6. Refer to the Channel Token value in the API Information section.

Additional Requirements

Other OCE components for Oracle Visual Builder applications may have additional requirements. Detailed component configuration instructions, including requirements and options, are found in the Read Me file of each component.

Add a Component to a Visual Builder Application Page

To add a component to an Oracle Visual Builder application page, you will need to first install the OCE component from a component exchange connected to your Oracle Visual Builder instance. For information on how to connect a component exchange,

see [Add a Connection to the Component Exchange](#) in the *Administering Oracle Visual Builder* guide.

Once an OCE component has been installed from the component exchange, click-and-drag the component from the component tab of the Oracle Visual Builder page designer into the desired slot on the page of the application and configure the component as required. For example, some components require an asset ID from Oracle Content Management. Some require a content type or layout.



Note:

Detailed information about what a component requires and how to configure it can be found in each component Read Me file once Oracle Visual Builder is installed and you're connected to the component exchange.

The following components are designed to integrate Oracle Content Management features into your Visual Builder applications and are available in the public component exchanges.

Component	Description
Content Item	Use a content item component to add a specific content item to an application page.
Content List	Use a content list component to dynamically display content items based on various filter criteria.
Content Search	Use a content search component to allow the user to dynamically control what items are shown for a content list component, and what happens when the results are selected. For example, insert a customized search bar to change or refresh the content that's displayed on an application page, or choose another action, such as opening a search results popup.
Content Place Holder	Use a content placeholder component to dynamically display a content item of one or more types. For example, use a content item placeholder on a designated detail page so that when a user clicks a link to get more detailed information for a particular content item, it will automatically load the detail view for the associated content item.
Image	Use an image component to display a single approved and published Oracle Content Management image.
Gallery	Use a gallery component to present a set of approved and published Oracle Content Management images.
Asset Uploader	Use an asset uploader component to offer a simplified way to upload documents from a local machine and add them to a chosen Oracle Content Management asset repository.

Embed Oracle Visual Builder Applications in an Oracle Content Management Site Page

You can embed a Visual Builder Cloud Service (VBCS) visual app or VBCS page in an Oracle Content Management site page.

- [Embed a VBCS Visual App in an Oracle Content Management Page](#)
- [Embed a VBCS Page in a Site Page](#)

Embed a VBCS Visual App in an Oracle Content Management Page

To embed a visual app in a site page, you need to create the app in VBCS and then add the app to the page.

Create a Visual App in VBCS

1. Connect to a VBCS Server.

For example: `https://vbcserver/ic/builder` (for a Universal Credits subscription with Oracle Content Management) or `https://vbcserver` (for a non-metered subscription) .

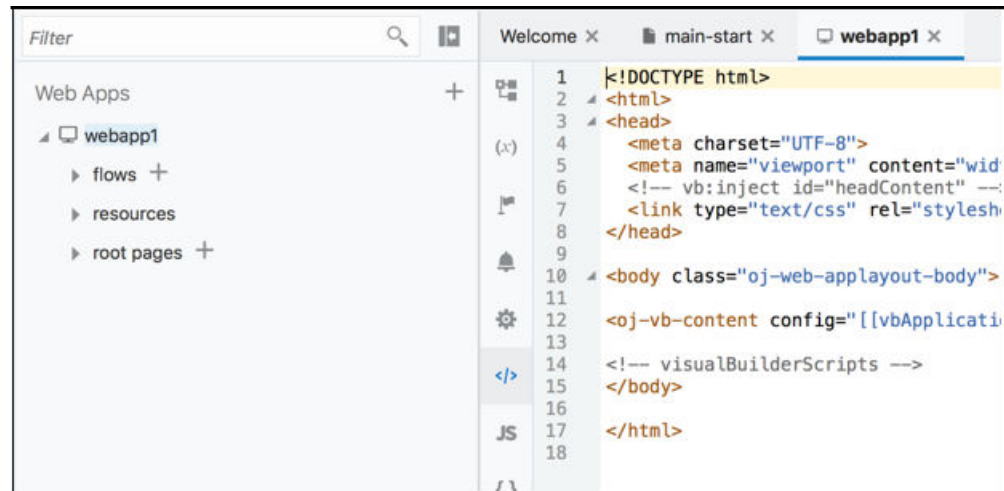
If you have a Universal Credits subscription, you must include `ic/builder` in your Service URL.

2. Allow Cross-Origin Resource Sharing (CORS):
 - a. From the VBCS menu, choose **Settings** and then **Allowed Origins**.
 - b. Click **New Origin** and enter the URL of your Oracle Content Management server for Origin Address.
3. Create a new Visual App in VBCS.
4. Create a web application.
 - a. Click **Web Applications** in the navigation menu on the left.
 - b. Enter a name for the web app, and then click **Create**.
5. Allow the web app to be embedded in an iframe:
 - a. Select the web app in the navigator.
 - b. Choose **Settings** (the cog icon), and then click the **Security** tab. Choose **Allow embedding in any application domain**.
6.
 - a. Right-click the **Resources** node in the navigator.
 - b. Locate the Sites SDK (import the `sites.js` or `sites.min.js` file).

The Sites SDK is also available for download from the Oracle Content Management server:

```
http://{server}/_sitesclouddelivery/renderer/app/sdk/js/  
sites.min.js
```

- c. Click **Import**. This imports the JS file into the resources directory.
7. Reference the Sites SDK in the page:
 - a. With the web app selected, choose the **HTML** tab in Site Builder.

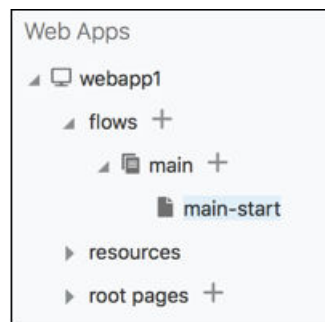


- b. Add the following line below the `<link>` tag:

```
<script type="text/javascript" src="resources/sites.min.js"></script>
```

- 8. Add a page URL parameter called `id`. Oracle Content Management will use this parameter to pass the ID of the component.

- a. Select the page in the web app.



- b. Click the **Variables** tab.
- c. Add a variable called `id`, and click **Create**.
- d. In the panel on the right, mark the new variable as a URL input parameter.
- 9. Add code to automatically set the iframe height when the web app renders:
 - a. Click the **JS** tab on the left of the page.

```

1  define([], function() {
2    'use strict';
3
4    var PageModule = function PageModule() {};
5
6    return PageModule;
7  });
8

```

The screenshot shows a code editor with three tabs: 'Welcome', 'main-start', and 'webapp1'. The 'main-start' tab is active, displaying the above JavaScript code. A 'Functions' button is visible in the bottom left corner of the editor.

- b. Add the following code above the `return` statement. This will resize the height of the Oracle Content Management component when the app renders.

```

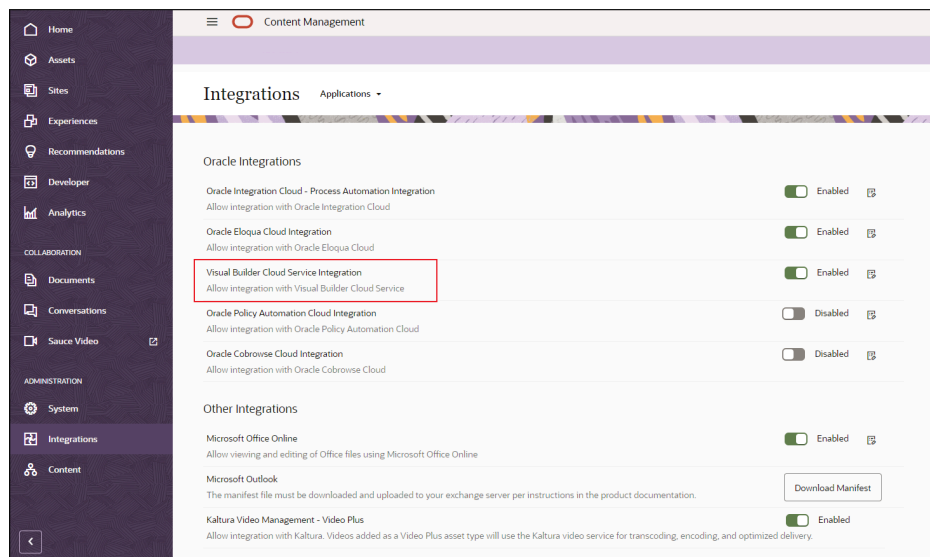
setTimeout(function() {
  SitesSDK.setProperty("height", null);
}, 500);

```

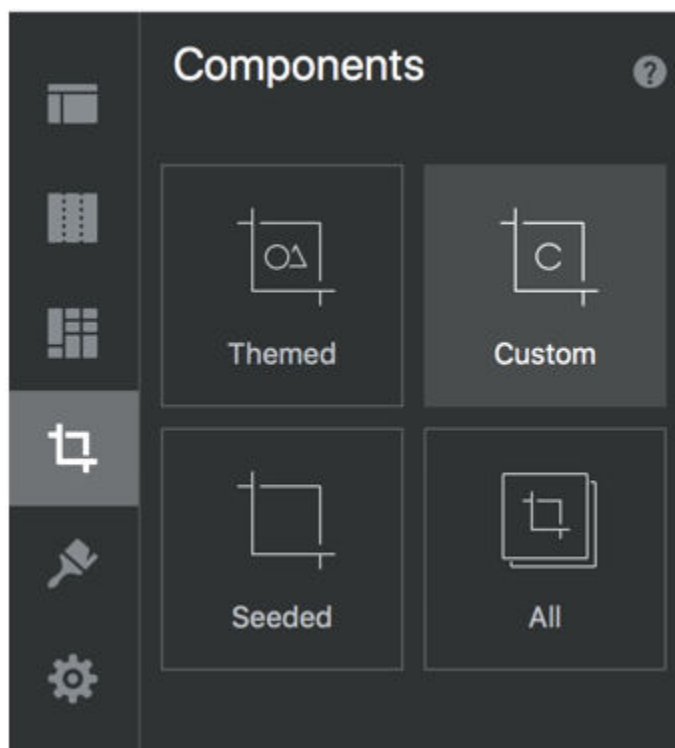
10. Stage and Publish the VBCS app. The app must be live for the Oracle Content Management site to use it.

Add the VBCS Visual App to an Oracle Content Management Site Page

1. In Oracle Content Management, configure a VBCS connection:
 - a. On the **Administration** menu, choose **Integrations** and then **Applications**.
 - b. Enable the **Visual Builder Cloud Service Integration**.



- c. Enter the URL of your Oracle Content Management instance, and then click **Save**.
2. Create a new VBCS component:
 - a. Choose **Developer** and then **Components**.
 - b. Choose **Create** and then **Create Visual Builder Component**.
 - c. Publish the VBCS app you created, copy the URL of the app, and then paste it into the form. Do the same for the web application you created.
3. Add the VBCS component to a site page:
 - a. Edit a new or existing site.
 - b. In Site Builder, choose **Components** and then **Custom**.



- c. Drag the VBCS component onto the site page.

Embed a VBCS Page in a Site Page

To embed a VBCS page in a site page, you need to create an app in VBCS and then add the app to the site page.

Create an App in VBCS

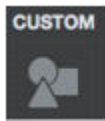
1. Connect to a VBCS Server.

For example: <https://vbcserver/ic/builder> (for a Universal Credits subscription with Oracle Content Management) or <https://vbcserver> (for a non-metered subscription) .

 **Note:**

If you have a Universal Credits subscription, you must include `ic/builder` in your Service URL.

2. Allow Cross-Origin Resource Sharing (CORS):
 - a. Choose **Administer Visual Builder**, then **Global Settings**, and then **Allowed Origins**.
 - b. Click **New Origin**, and enter the URL of your Oracle Content Use Oracle Content Management Components in Oracle Visual Builder Applications server for Origin Address.
3. Create a new app in VBCS.
4. Allow the new app to be embedded.
 - a. Choose **Application Settings**, then **Security**, and then **Embedding**.
 - b. Select **Allow embedding in any domain**.
5. Use Data Designer and Page Designer to build your app.
6. Add a custom component to the bottom of your page.



7. Select the Custom Component. Enter the following in the **Template** section:

```
<div data-bind="html: script"></div>
```

8. Enter the following code in the **Model** section, substituting your own Oracle Content Management server.

```
define([], function () {
    'use strict';

    /**
     * Inject the SitesSDK and set the Component Height.
     */
    var CustomComponentViewModel = function (params, componentInfo)
    {
        this.script = ko.observable(
            '<script type="text/javascript">\n'
            + '(function(d, s, id) {\n'
            + '    var js, fjs = d.getElementsByTagName(s)[0];\n'
            + '    if (d.getElementById(id))\n'
            + '        return;\n'
            + '    js = d.createElement(s);\n'
            + '    js.id = id;\n'
            + '    js.src = "https://oracle-content-management-server
```



```
        + ' fjs.parentNode.insertBefore(js, fjs);\n'
        + ' }(document, "script", "sites-sdk");\n'
        + ' setTimeout(function() {\n'
        + '     SitesSDK.setProperty("height", null);\n'
        + ' }, 500);\n'
    + '</script>');
};

    return CustomComponentViewModel;
});
```

9. Enter the following for **Application Style**. This will hide some of the unwanted "chrome" around the component when embedded in an SCS page.

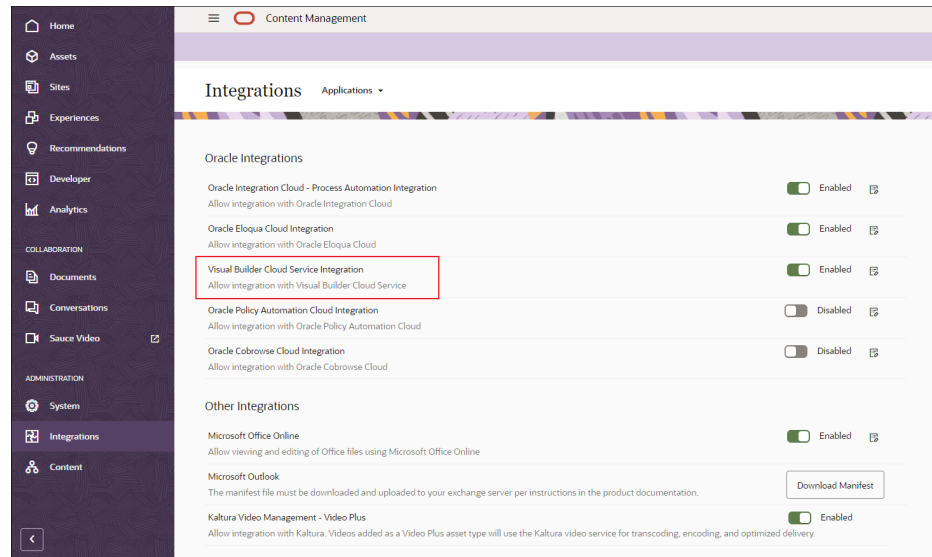
```
/* remove some side padding */
div#abcs-app-content > div {
    max-width: none;
}

/* allow SitesSDK.setProperty("height") to work */
html, body, body.abcs-layout-nonnav {
    height: auto;
}
```

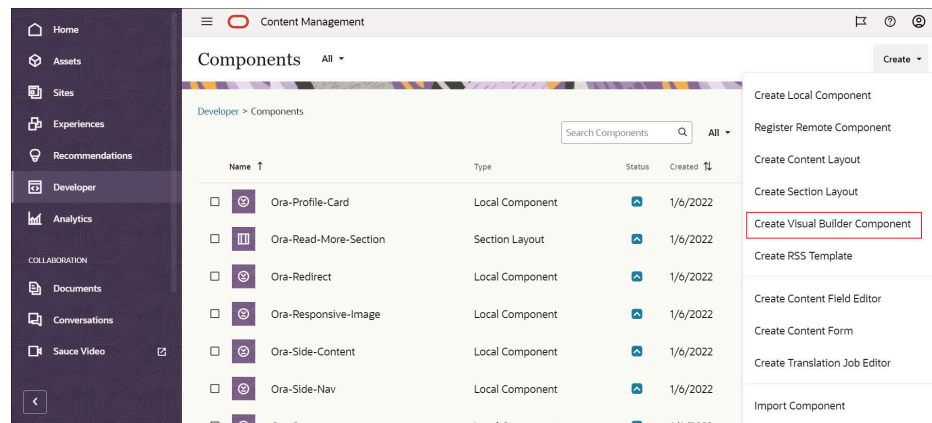
10. Stage and publish the VBCS app. The app must be live for Oracle Content Management sites to use it.

Add the VBCS App to a CECS Site Page

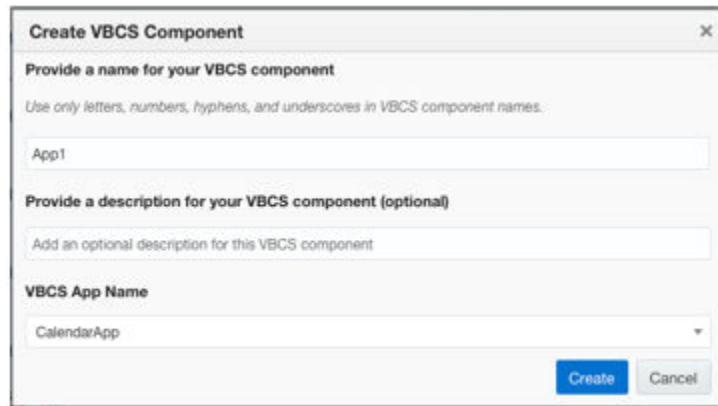
1. In Oracle Content Management, configure a VBCS Connection:
 - a. Choose **Administration, Integrations**, and then **Oracle Integrations**.
 - b. Select **Enable** for **Visual Builder Cloud Service Integration**.
 - c. Enter the Oracle Content Management URL (from Step 1 under Create an App in VBCS), and then click **Save**.



2. Create a new VBCS Component:
 - a. Choose **Developer** and then **Components**,
 - b. Choose **Create** and then **Create Visual Builder Component**.



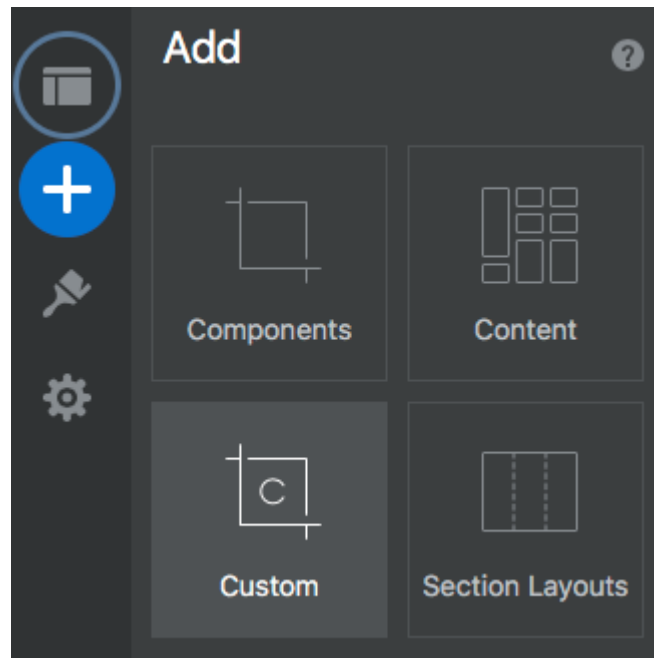
- c. In the drop-down list, choose the VBCS App (created earlier in step 3 under Create an App in VBCS).



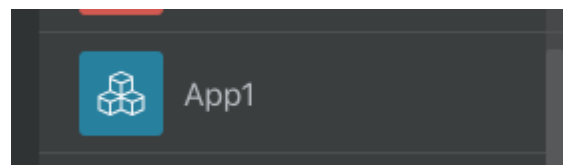
The dialog box is titled "Create VBCS Component" and contains the following fields and buttons:

- Provide a name for your VBCS component**: A text input field containing "App1". Below it is a note: "Use only letters, numbers, hyphens, and underscores in VBCS component names."
- Provide a description for your VBCS component (optional)**: A text input field containing "Add an optional description for this VBCS component".
- VBCS App Name**: A dropdown menu with "CalendarApp" selected.
- Buttons: "Create" (blue) and "Cancel" (grey).

3. Add the VBCS component to a site page.
 - a. Edit a new or existing site .
 - b. In Site Builder, choose **Add** and then **Custom**.



- c. Drag the VBCS component onto the page.



Build an Oracle Content Management VBCS Form and Data Report Components

You can build local Oracle Content Management components that use REST APIs exposed by business objects in VBCS to deliver a variety of forms.

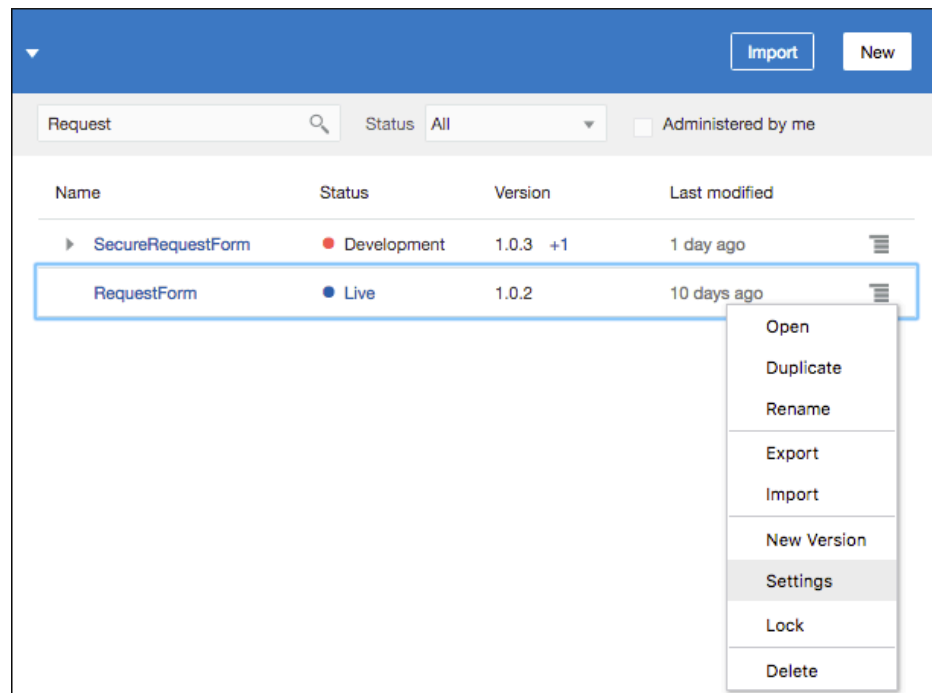
- [Build an Oracle Content Management VBCS Public Form Component](#)
- [Build an Oracle Content Management VBCS Secure Form Component](#)
- [Build an Oracle Content Management VBCS Public Gated Form Component](#)

Build an Oracle Content Management VBCS Public Form Component

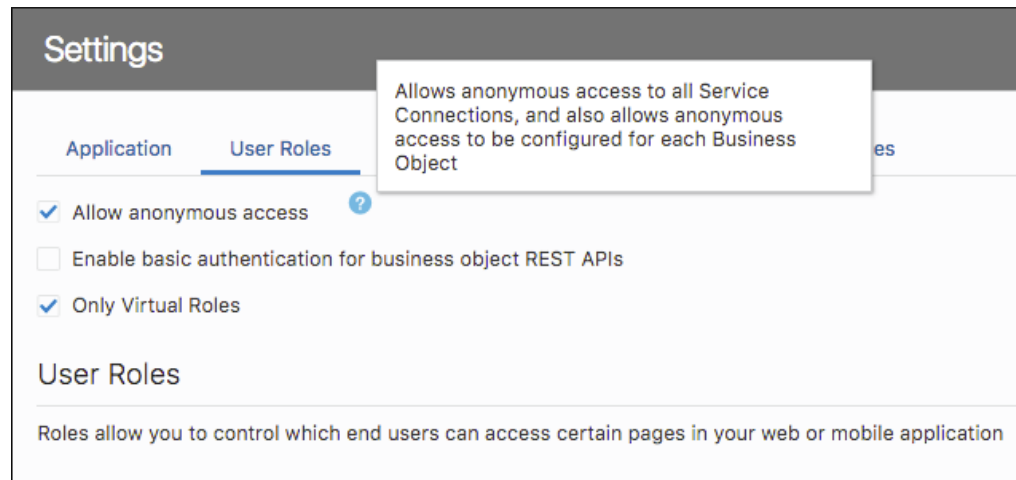
You can build a local Oracle Content Management component that uses REST APIs exposed by business objects in VBCS to deliver a simple, anonymous, public form.

VBCS Configuration

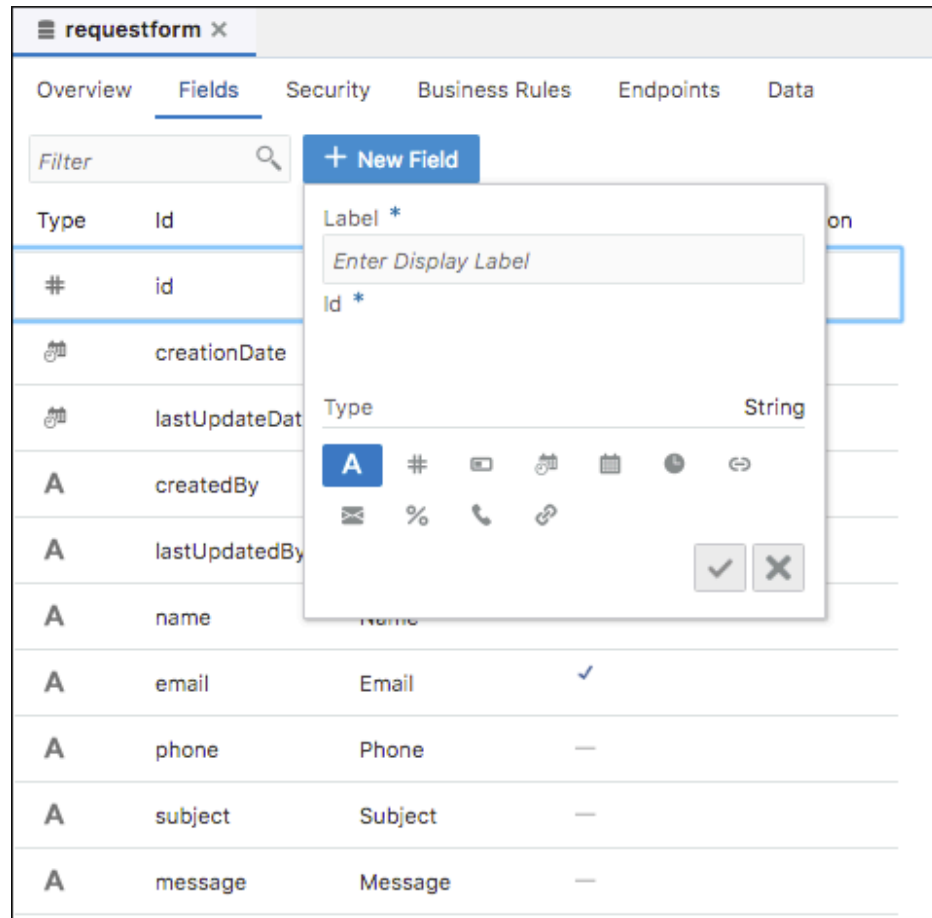
1. Allow Cross-Origin Resource Sharing (CORS):
 - a. Choose **Visual Builder**, then **Settings**, and then **Allowed Origins**.
 - b. Click **New Origin** and enter the URL of your Oracle Content Management server for Origin Address.
 - c. Click the check mark to save.
2. Create a new Application:
3. Configure the app to allow anonymous access.
 - a. Open Application Settings.



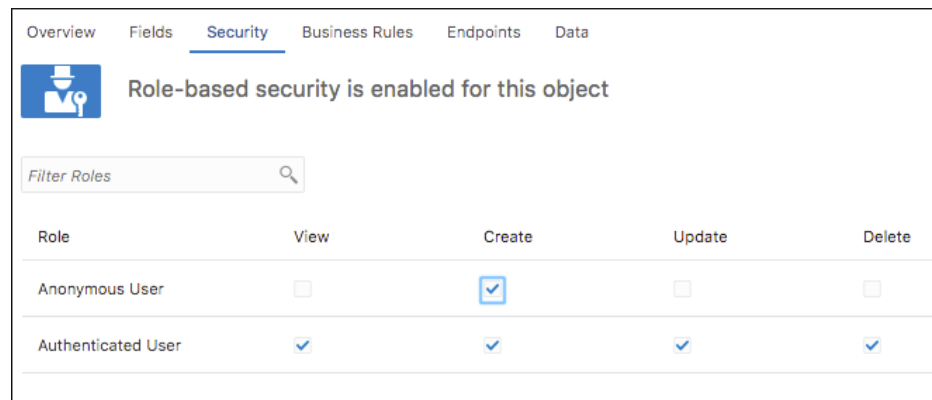
- b. On the Settings page, choose **User Roles**.



- c. Select **Allow anonymous access**.
4. Create a business object:
 - Add fields.



- Enable role-based security.
- Grant Anonymous User the Create permission.



Build an Oracle Content Management Local Component

Assumptions:

- The VBCS app name is "RequestForm".

- The business object name is "requestform" and it contains the following custom fields:
 - name (required)
 - email (required)
 - phone
 - subject
 - message

Modify assets/render.js

1. Define the component template as follows.

```

<!-- ko if: initialized -->
<div class="form">

    <!-- ko if: requestSuccessMsg -->
    <div class="request-msg green" data-bind="text:
requestSuccessMsg"></div>
    <!-- /ko -->
    <!-- ko if: requestFailMsg -->
    <div class="request-msg red" data-bind="text: requestFailMsg"></
div>
    <!-- /ko -->

    <label class="required-field" for="name">Name</label>
    <input type="text" id="name" name="name" required
placeholder="Your name. . ." data-bind="value: name"/>

    <label class="required-field" for="email">Email</label>
    <input type="text" id="email" name="email" required
placeholder="Your email. . ." data-bind="value: email"/>

    <label for="phone"></label>Phone</label>
    <input type="text" id="phone" name="phone" data-bind="value:
phone"/>
    <label for="subject">Subject</label>
    <input type="text" id="subject" name="subject" data-bind="value:
subject"/>

    <label for="message"></label>Message</label>
    <textarea id="message" name="message" rows="6" data-bind="value:
message"/>

    <button data-bind="click: sendRequest, , enable: canSubmit">Send
Request</button></div>

<!-- note that the component has completed rendering into the page -->
<div class="scs-hidden" data-bind="scsRenderStatus: {'id': id, 'status':
'complete'}"></div>
<!-- /ko -->

```

2. Create the observables for the fields in the Knockout ViewModel.

```

self.initialized = ko.observable(false);
self.requestSuccessMsg = ko.observable();
self.requestFailMsg = ko.observable();
self.VBCSServerUrl = ko.observable();
self.name = ko.observable();
self.email = ko.observable();
self.phone = ko.observable();
self.subject = ko.observable();
self.message = ko.observable();

// Get VBCS server
var serverPromise = getVBCSServerURL();
    serverPromise.then(function (result) {
        self.VBCSServerUrl(result.url);
        self.initialized(true);
    });

self.canSubmit = ko.computed(function () {
    return self.name() && self.email();
}, self);

```

3. Handle required fields.

Enable the Submit button only after all required fields have values.

4. Obtain the VBCS connection.

After you configure VBCS connection, there are two ways to get the connection:

- From siteinfo at site runtime
- From Integrations in Site Builder

```

var getVBCSServerURL = function () {
    var serverPromise = new Promise(function (resolve, reject) {
        // First try to get from siteinfo
        var siteConnections =
SCSRenderAPI.getSiteProperty('siteConnections');
        var serverUrl = siteConnections &&
siteConnections.VBCSConnection;
        if (serverUrl) {
            console.log('Get VBCS server from siteinfo: ' + serverUrl);
            resolve({'url': serverUrl});
        } else {
            // Get from integrations
            var configUrl = '/documents/web?
IdcService=AF_GET_APP_INFO_SIMPLE&dAppName=VBCS';
            $.ajax({
                type: 'GET',
                dataType: 'json',
                url: configUrl,
                success: function (data) {

```



```

        var appInfo = data.ResultSets.AFApplicationInfo;
        var enabled;
        if (appInfo) {
            for (var i = 0; i < appInfo.fields.length; i +=
1) {
                if (appInfo.fields[i].name ===
'dAppEndPoint') {
                    serverUrl =
appInfo.rows[appInfo.currentRow][i];
                } else if (appInfo.fields[i].name ===
'dIsAppEnabled') {
                    enabled = appInfo.rows[appInfo.currentRow]
[i];
                }
                if (serverUrl && enabled) {
                    break;
                }
                if (enabled !== '1') {
                    serverUrl = '';
                }
            }
            console.log('Get VBCS server from Idc Service: ' +
serverUrl);
            resolve({'url': serverUrl});
        },
        error: function (xhr, status, err) {
            console.log('Request failed: url:' + configUrl + '
status: ' + status + ' error: ' + err);
            resolve({'url': serverUrl});
        }
    });
    });
    return serverPromise;
};

```

5. Submit the request

```

self.sendRequest = function (data, event) {
    var vbcsServer = self.VBCSServerUrl();
    var appName = 'requestform',
        appVersion = 'live',
        businessObject = 'Requestform'
    var url = vbcsServer + '/rt/' + appName + '/' + appVersion + '/'
resources/data/' + businessObject;
    var payload = {
        "name": self.name(),
        "email": self.email(),
        "phone": self.phone(),
        "subject": self.subject(),
        "message": self.message()
    };
    $.ajax({
        type: 'POST',

```

```
        url: url,
        beforeSend: function(xhr) {
            xhr.setRequestHeader( "Content-type", "application/
vnd.oracle.adf.resourceitem+json" );
        },
        data: JSON.stringify(payload),
        dataType: 'json',
        success: function (data) {
            self.requestFailMsg('');
            self.requestSuccessMsg('Request has been submitted
successfully');
            self.name('');
            self.email('');
            self.phone('');
            self.subject('');
            self.message('');
        },
        error: function(jqXHR, textStatus, errorThrown) {
            console.log('Error:');
            console.log(jqXHR);
            self.requestSuccessMsg('');
            self.requestFailMsg('Failed to submit the request');
        }
    });
};
```

Modify styles/design.css

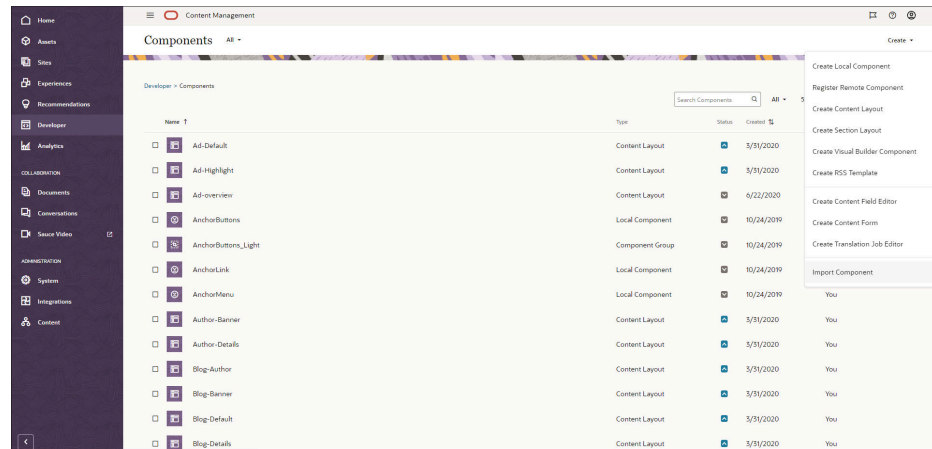
Add the following css to design.css.

```
.form {
    font-family: "Helvetica Neue", "Segoe UI", sans-serif-regular,
Helvetica, Arial, sans-serif;
    font-size: 14px;
}
.form input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
.form textarea {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
.form button {
```

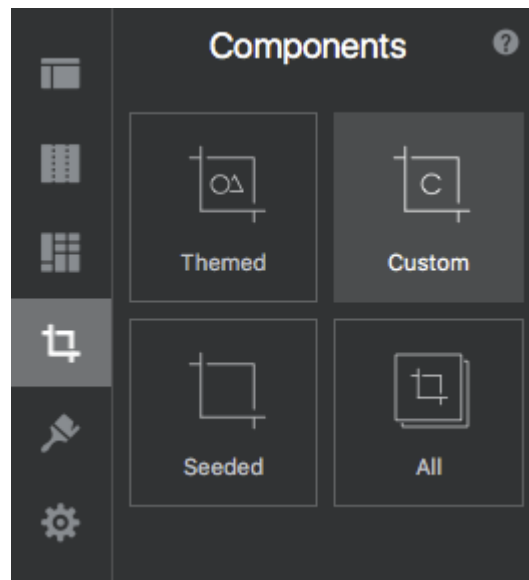
```
width: 100%;
background-color: #4CAF50;
color: white;
padding: 14px 20px;
margin: 8px 0;
border: none;
border-radius: 4px;
cursor: pointer;
}
.form button:hover {
    background-color: #45a049;
}
.form button:disabled {
    background-color: #dddddd;
}
.required-field::after {
    content: "*";
    color: red;
    margin-left: 2px;
}
.request-msg {
    padding: 5px;
    font-size: 18px;
    font-weight: bold;
    text-align: center;
    margin-bottom: 20px;
}
.green {
    background-color: #81BA5E;
}
.red {
    background-color: red;
}
```

Use the Form Component on Oracle Content Management

1. Configure the VBCS connection:
 - Choose **Administration**, then **Integrations**, and then **Applications**.
 - Click the **Visual Builder Cloud Service Integration** check box.
 - Enter the URL, and click **Save**.
2. Import the component:
 - Choose **Developer** and then **Components**.
 - Choose **Create** and then **Import Component**.



3. Add the component to a page:
 - a. Edit a new or existing site.
 - b. In Site Builder, choose **Components** and then **Custom**.



- c. Drag the component onto the page.

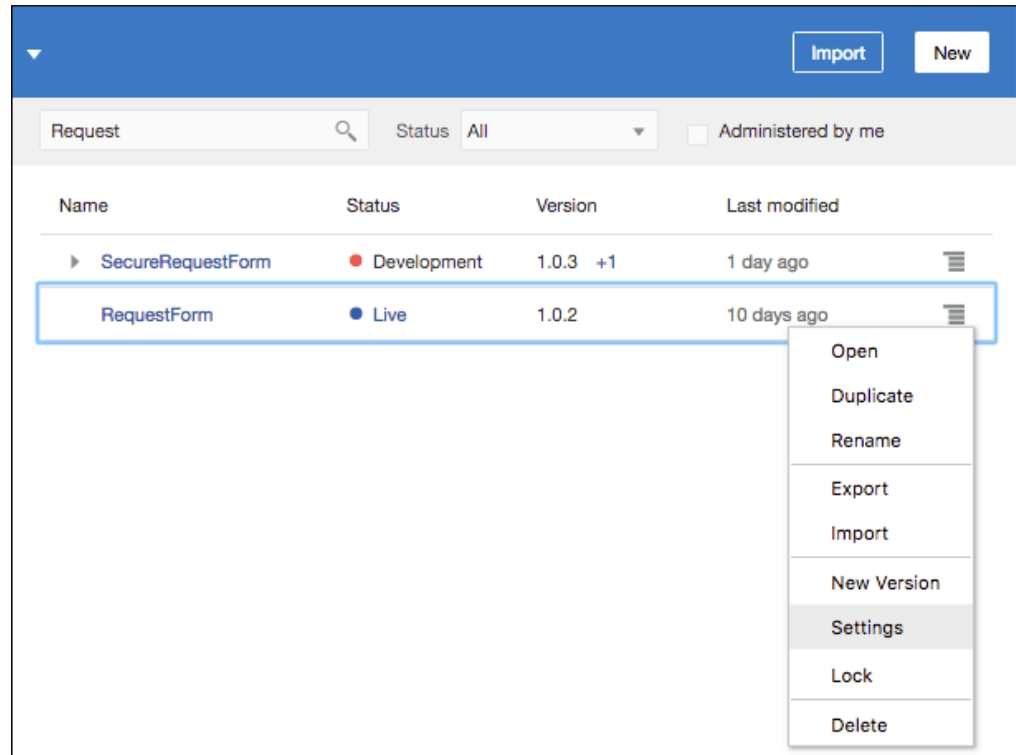
Build an Oracle Content Management VBCS Secure Form Component

You can build a local Oracle Content Management component that uses REST APIs exposed by business objects in VBCS to deliver a simple web form that requires user authentication.

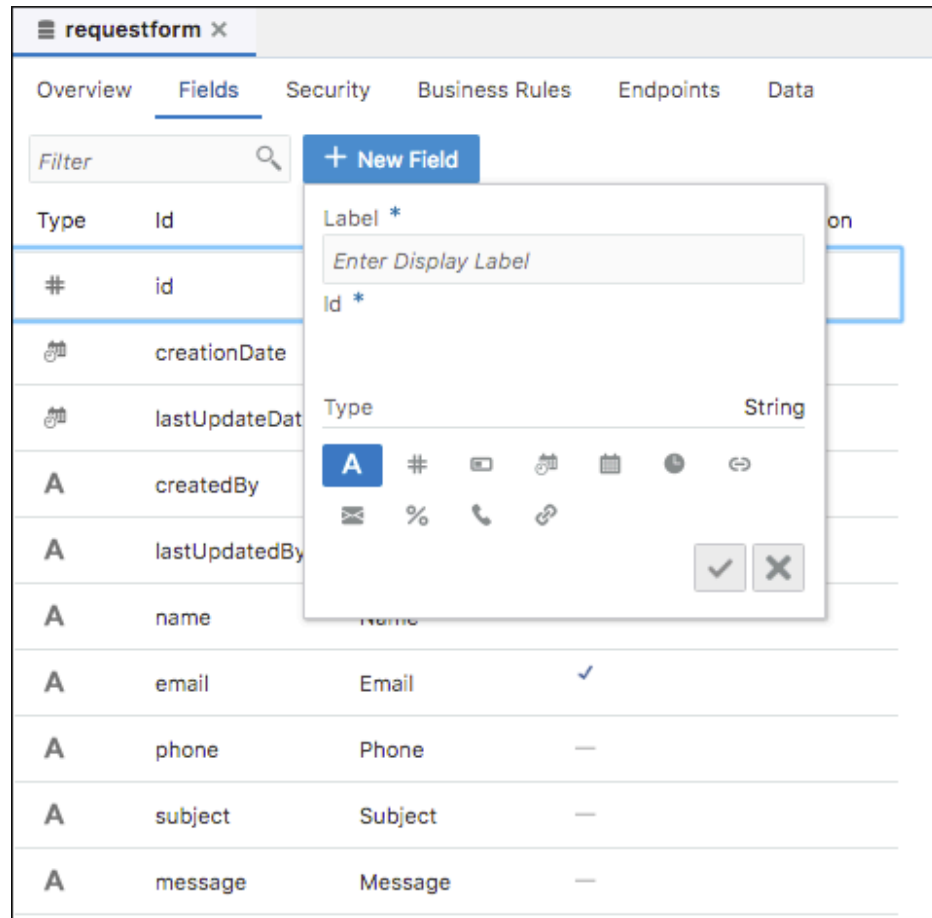
VBCS Configuration

1. Allow Cross-Origin Resource Sharing (CORS):
 - a. Choose **Visual Builder**, then **Settings**, and then **Allowed Origins**.

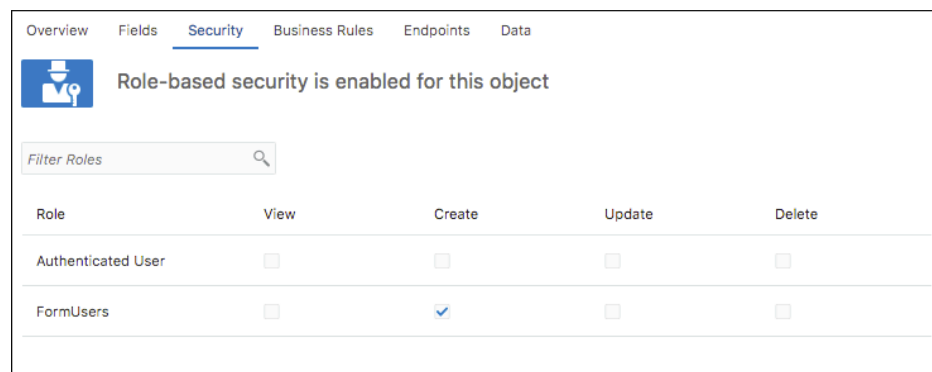
- b. Click **New Origin** and enter the URL of your Oracle Content Management server for Origin Address.
 - c. Click the check mark to save.
2. Create a new Application
3. Configure the app to allow access for authenticated users.
 - a. Open **Application Settings**.



- b. On the Settings page, choose **User Roles**.
 - c. Add roles to control access to the business object.
The values for **Mapping** are groups from Oracle Identity Cloud Service. To add groups, see Create Groups for Your Organization.
4. Create a business object.
 - Add fields.



- Enable role-based security
- Grant roles to Create permission.



Build an Oracle Content Management Local Component

Assumptions:

- The VBCS app name is "RequestForm".
- The business object name is "requestform" and it contains the following custom fields:

- name (required)
- email (required)
- phone
- subject
- message

Modify assets/render.js

1. Define the component template as follows.

```
<!-- ko if: initialized -->
<div class="form">

    <!-- ko if: requestSuccessMsg -->
    <div class="request-msg green" data-bind="text:
requestSuccessMsg"></div>
    <!-- /ko -->
    <!-- ko if: requestFailMsg -->
    <div class="request-msg red" data-bind="text: requestFailMsg"></
div>
    <!-- /ko -->

    <label class="required-field" for="name">Name</label>
    <input type="text" id="name" name="name" required
placeholder="Your name. . ." data-bind="value: name"/>

    <label class="required-field" for="email">Email</label>
    <input type="text" id="email" name="email" required
placeholder="Your email. . ." data-bind="value: email"/>

    <label for="phone"></label>Phone</label>
    <input type="text" id="phone" name="phone" data-bind="value:
phone"/>

    <label for="subject">Subject</label>
    <input type="text" id="subject" name="subject" data-bind="value:
subject"/>

    <label for="message"></label>Message</label>
    <textarea id="message" name="message" rows="6" data-bind="value:
message"/>

    <button data-bind="click: sendRequest, , enable: canSubmit">Send
Request</button>
</div>

<!-- note that the component has completed rendering into the page -->
<div class="scs-hidden" data-bind="scsRenderStatus: {'id': id, 'status':
'complete'}"></div>
<!-- /ko -->
```

2. Create the observables for the fields in the Knockout ViewModel.

```

self.initialized = ko.observable(false);
self.requestSuccessMsg = ko.observable();
self.requestFailMsg = ko.observable();
self.VBCSServerUrl = ko.observable();
self.name = ko.observable();
self.email = ko.observable();
self.phone = ko.observable();
self.subject = ko.observable();
self.message = ko.observable();

// Get VBCS server
var serverPromise = getVBCSServerURL();
    serverPromise.then(function (result) {
        self.VBCSServerUrl(result.url);
        self.initialized(true);
    });

self.canSubmit = ko.computed(function () {
    return self.name() && self.email();
}, self);

```

3. Handle required fields.

Enable the Submit button only after all required fields have values.

4. Obtain the VBCS connection

After configure VBCS connection, there are two ways to get the connection:

- From siteinfo at site runtime
- From Integrations in Site Builder

```

var getVBCSServerURL = function () {
    var serverPromise = new Promise(function (resolve, reject) {
        // First try to get from siteinfo
        var siteConnections =
SCSRenderAPI.getSiteProperty('siteConnections'
        var serverUrl = siteConnections &&
siteConnections.VBCSConnection;
        if (serverUrl) {
            console.log('Get VBCS server from siteinfo: ' +
serverUrl);
            resolve({'url': serverUrl});
        } else {
            // Get from integrations
            var configUrl = '/documents/web?
IdcService=AF_GET_APP_INFO_SIMPLE&dAppName=VBCS';
            $.ajax({
                type: 'GET',
                dataType: 'json',
                url: configUrl
                success: function (data) {

```



```

        var appInfo = data.ResultSets.AFApplicationInfo;
        var enabled;
        if (appInfo) {
            for (var i = 0; i < appInfo.fields.length; i
+= 1) {
                if (appInfo.fields[i].name ===
'dAppEndPoint') {
                    serverUrl =
appInfo.rows[appInfo.currentRow][i];
                } else if (appInfo.fields[i].name ===
'dIsAppEnabled') {
                    enabled =
appInfo.rows[appInfo.currentRow][i];
                }
                if (serverUrl && enabled) {
                    break;
                }
            }
            console.log('Get VBCS server from Idc Service: '
+ serverUrl);
            resolve({'url': serverUrl});
        },
        error: function (xhr, status, err) {
            console.log('Request failed: url:' + configUrl +
' status: ' + status + ' error: ' + err);
            resolve({'url': serverUrl});
        }
    });
    });
    return serverPromise;
};

```

5. Get an authorization token.

Requirement: Oracle Content Management and VBCS are deployed in the same identity domain.

```

var getAuthToken = function (args) {
    // dummy function if callbacks not supplied
    var dummyCallback = function () {};

    // extract the args and create the server URL
    var serverURL = (args.serverURL || '/').split('/ic/')[0],
        successCallback = args.successCallback || dummyCallback,
        errorCallback = args.errorCallback || dummyCallback,
        tokenURL = serverURL + '/ic/builder/resources/security/token';

    // For VBCS to get the authtoken:
    // - make a POST call to /ic/builder/resources/security/token
    // - include scope=run-time form parameter
    var getToken = function (tokenURL, successCallback, errorCallback) {
        $.ajax({
            'type': 'POST',
            'url': tokenURL,

```

```

        data: {
            scope: 'run-time'
        },
        'xhrFields': {
            withCredentials: true
        },
        'success': successCallback
    }).fail(errorCallback);
};

// try to get the token normally
getToken(tokenURL,
    function (resp, status, xhr) {
        var ct = xhr.getResponseHeader("content-type") || "";

        // if the response was an HTML Form. . .
        if (ct.indexOf('html') > -1) {
            // parse the form and submit it
            var parser = new DOMParser(),
                htmlDoc = parser.parseFromString(resp, "text/
html"),

                forms = htmlDoc.getElementsByTagName("form");
            if (forms.length === 1) {
                var f = forms[0];
                $.ajax({
                    'type': 'POST',
                    'url': f.action,
                    'data': $(f).serialize(),
                    'xhrFields': {
                        'withCredentials': true
                    },
                    'success': function () {
                        // retry getting the token now the form
                        was auto-submitted
                        getToken(tokenURL, successCallback,
                        errorCallback);
                    }
                }).fail(function () {
                    // even if the form submit failed, retry
                    getting the token
                    getToken(tokenURL, successCallback,
                    errorCallback);
                });
            }
        } else {
            // already logged in return the token
            successCallback(resp);
        }
    },
    errorCallback);
};

```

6. Submit the request.

```

self.sendRequest = function (data, event) {
    var vbcsServer = self.VBCSServerUrl();

```

```
var authorization, token;
var appName = 'securerequestform',
    mode = 'rt',
    appVersion = 'live',
    businessObject = 'Requestform';
var url = vbcsServer + '/' + mode + '/' + appName + '/' + appVersion
+ '/resources/data/' + businessObject;
var payload = {
  "name": self.name(),
  "email": self.email(),
  "phone": self.phone(),
  "subject": self.subject(),
  "message": self.message()
};
// get token first
getAuthToken({
  'serverURL': self.VBCSServerUrl(),
  'successCallback': function (data) {
    token = data;
    authorization = (token.token_type ? token.token_type :
'Bearer') + ' ' + token.access_token;
    $.ajax({
      type: 'POST',
      url: url,
      beforeSend: function (xhr) {
        xhr.setRequestHeader('Content-type', 'application/
vnd.oracle.adf.resourceitem+json');
        xhr.setRequestHeader('Authorization', authorization);
      },
      data: JSON.stringify(payload),
      dataType: 'json',
      success: function (data) {
        self.requestFailMsg('');
        self.requestSuccessMsg('Request has been submitted
successfully');
        self.name('');
        self.email('');
        self.phone('');
        self.subject('');
        self.message('');
      },
      error: function (jqXHR, textStatus, errorThrown) {
        console.log('Error:');
        console.log(jqXHR);
        self.requestSuccessMsg('');
        self.requestFailMsg('Failed to submit the request');
      }
    });
  },
  'errorCallback': function (xhr, status, err) {
    if (xhr && xhr.status === 200) {
      token = xhr.responseText;
      console.log('Got token');
    } else {
      console.error('getToken: xhr: ' + JSON.stringify(xhr) + '');
```

```
status: ' + status + ' error: ' + err);
        self.requestSuccessMsg('');
        self.requestFailMsg('Failed to get authorization
token');
    }
}
});
};
```

Modify styles/design.css

Add the following css to design.css.

```
.form {
    font-family: "Helvetica Neue", "Segoe UI", sans-serif-regular,
Helvetica, Arial, sans-serif;
    font-size: 14px;
}
.form input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
.form textarea {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
.form button {
    width: 100%;
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
.form button:hover {
    background-color: #45a049;
}
.form button:disabled {
    background-color: #dddddd;
}
.required-field::after {
    content: "*";
    color: red;
}
```

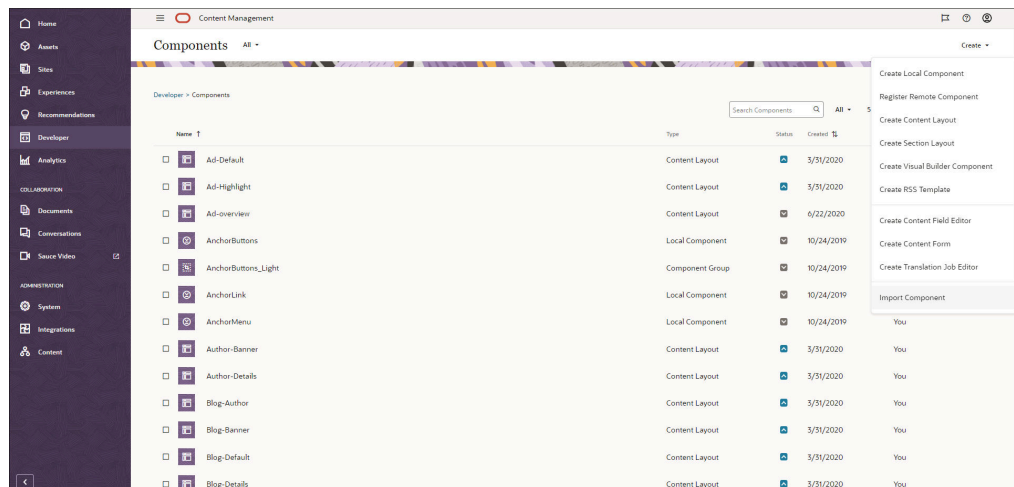
```

margin-left:2px
}
.request-msg {
padding: 5px;
font-size: 18px;
font-weight: bold;
text-align: center;
margin-bottom: 20px;
}
.green {
background-color: #81BA5E;
}
.red {
background-color: red;
}

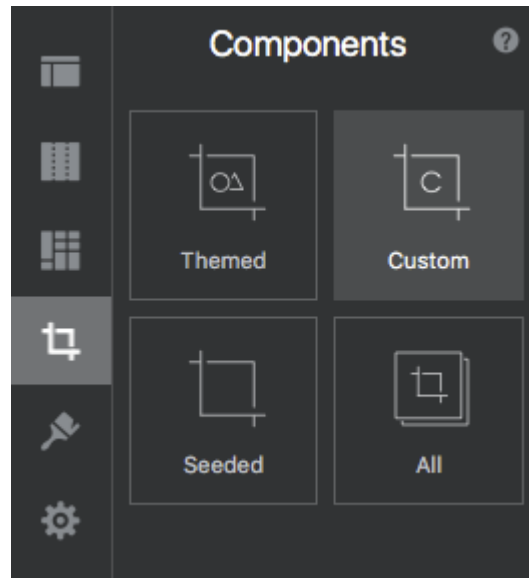
```

Use the Form Component on Oracle Content Management

1. Configure the VBCS connection:
 - Choose **Administration**, then **Integrations**, and then **Applications**.
 - Click the **Visual Builder Cloud Service Integration** check box.
 - Enter the URL, and click **Save**.
2. Import the component:
 - Choose **Developer** and then **Components**.
 - Choose **Create** and then **Import Component**.



3. Add the component to a page
 - a. Edit a new or existing site.
 - b. In Site Builder, choose **Components** and then **Custom**.



- c. Drag the component onto the page.

 **Note:**

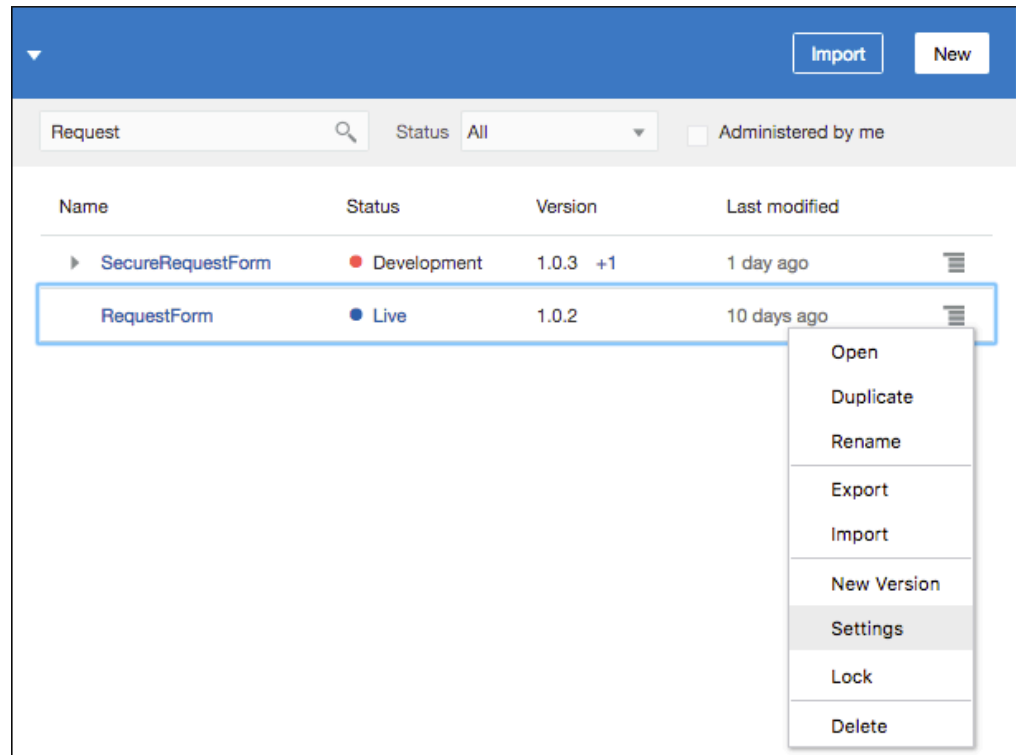
This VBCS secure form component works only on secure sites.

Build an Oracle Content Management VBCS Public Gated Form Component

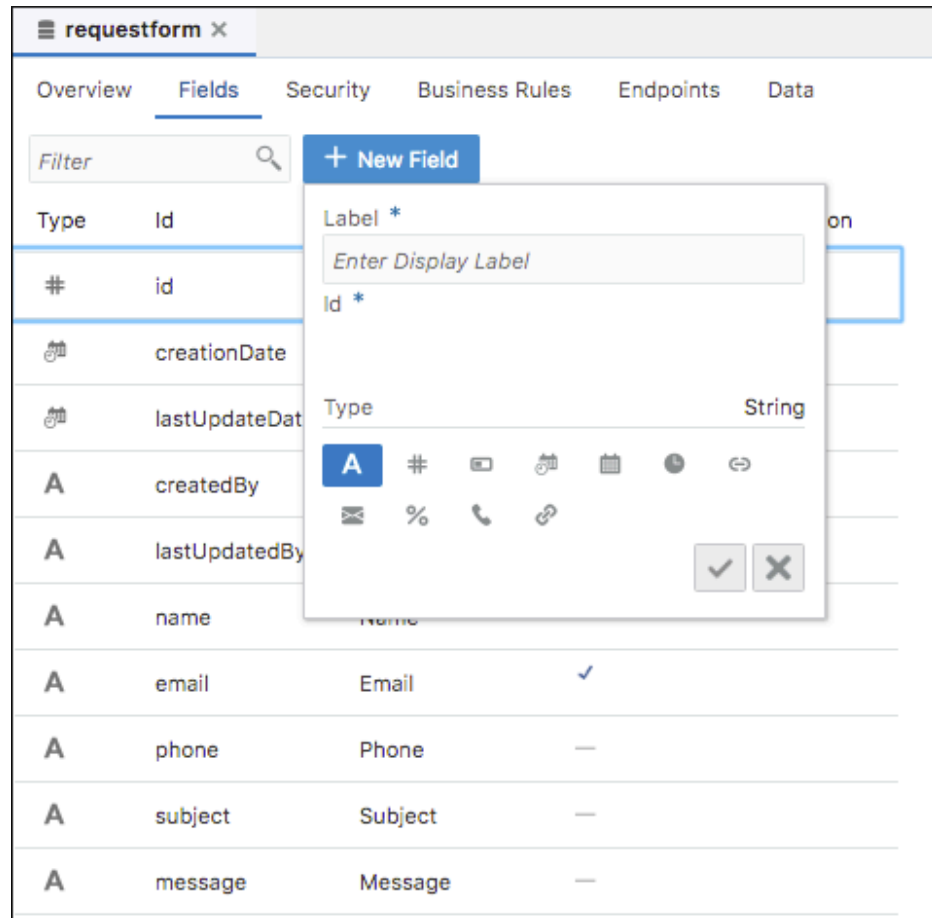
You can build a local Oracle Content Management component that uses REST APIs exposed by business objects in VBCS to deliver a simple, anonymous web form that captures visitor details to allow the visitor to download a document.

VBCS Configuration

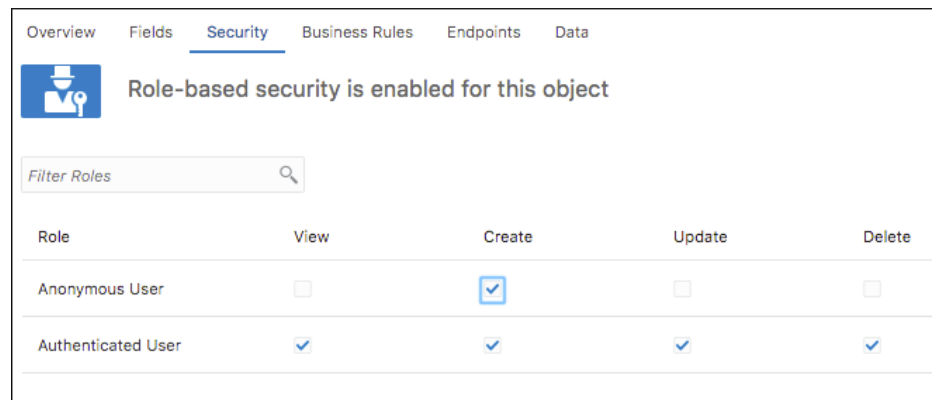
1. Allow Cross-Origin Resource Sharing (CORS):
 - a. Choose **Visual Builder** , then **Settings**, and then **Allowed Origins**.
 - b. Click **New Origin** and enter the URL of your Oracle Content Management server for Origin Address.
 - c. Click the check mark to save.
2. Create a new Application.
3. Configure the app to allow anonymous access.
 - a. Open Application Settings .



- b. On the Settings page, choose **User Roles**.
 - c. Select **Allow anonymous access**.
4. Create a business object:
- Add fields.



- Enable role-based security.
- Grant Anonymous User the Create permission.



Build an Oracle Content Management Local Component

Assumptions:

- The VBCS app name is "RequestForm".
- The business object name is "registration" and it contains the custom fields.

- firstName (required)
- lastName (required)
- email (required)
- phone
- company
- jobTitle

Modify assets/render.js

1. Define the component template as follows.

```
<!-- ko if: initialized -->

div class="form">
  <!-- ko if: !showDownload() -->
  <h1 style="text-align: center;">Fill out the form to access the
document</h1>

  <!-- ko if: requestSuccessMsg -->
  <div class="request-msg green" data-bind="text:
requestSuccessMsg"></div>
  <!-- /ko -->
  <!-- ko if: requestFailMsg -->
<div class="request-msg red" data-bind="text: requestFailMsg"></div>
<!-- /ko -->

  <label class="required-field" for="firstname">First Name</label>
  <input type="text" id="firstname" name="firstname" required
placeholder="Your first name. . ." data-bind="value: firstName"/>
  <label class="required-field" for="lastname">Last Name</label>
  <input type="text" id="lastname" name="lastname" required
placeholder="Your last name. . ." data-bind="value: lastName"/>
  <label class="required-field" for="email">Business E-mail</label>
  <input type="text" id="email" name="email" required
placeholder="Your email. . ." data-bind="value: email"/>
  <label for="phone"></label>Phone</label>
  <input type="text" id="phone" name="phone" data-bind="value:
phone"/>
  <label for="company">Company</label>
  <input type="text" id="company" name="company" data-bind="value:
company"/>
  <label for="jobtitle"></label>Job Title</label>
  <input type="text" id="jobtitle" name="jobtitle" data-bind="value:
jobTitle"/>

  <button data-bind="click: sendRequest, enable: canSubmit">Accept
the document</button>
  <!-- /ko -->

  <!-- ko if: showDownload() -->
  <div class="download">
    <h2>Thanks for your registration. Please click the button
to download. </h2>
```

```

        <button data-bind="click: startDownload">Download</
button>
        <button data-bind="click: closeDownload">Close</
button>
    </div>
    <!-- /ko -->
</div>

<!-- note that the component has completed rendering into the page
-->
<div class="scs-hidden" data-bind="scsRenderStatus: {'id': id,
'status': 'complete'}"></div>
<!-- /ko -->

```

2. Create the observables for the fields in the Knockout ViewModel.

```

self.initialized = ko.observable(false);
self.requestSuccessMsg = ko.observable();
self.requestFailMsg = ko.observable();
self.VBCSServerUrl = ko.observable();
self.showDownload = ko.observable(false);

self.firstName = ko.observable();
self.lastName = ko.observable();
self.email = ko.observable();
self.phone = ko.observable();
self.company = ko.observable();
self.jobTitle = ko.observable();

self.canSubmit = ko.computed(function () {
    return self.firstName() && self.lastName() && self.email();
}, self);

```

3. Handle required fields.

Enable the **Submit** button only after all required fields have values.

4. Obtain the VBCS connection.

After you configure a VBCS connection, there are two ways to get the connection:

- From siteinfo at site runtime
- From **Integrations** in Site Builder

```

var getVBCSServerURL = function () {
    var serverPromise = new Promise(function (resolve, reject) {
        // First try to get from siteinfo
        var siteConnections =
SCSRenderAPI.getSiteProperty('siteConnections');
        var serverUrl = siteConnections &&
siteConnections.VBCSConnection;
        if (serverUrl) {
            console.log('Get VBCS server from siteinfo: ' + serverUrl);

```

```

        resolve({'url': serverUrl});
    } else {
        // Get from integrations
        var configUrl = '/documents/web?
IdcService=AF_GET_APP_INFO_SIMPLE&dAppName=VBCS';
        $.ajax({
            type: 'GET',
            dataType: 'json',
            url: configUrl,
            success: function (data) {
                var appInfo = data.ResultSets.AFApplicationInfo;
                var enabled;
                if (appInfo) {
                    for (var i = 0; i < appInfo.fields.length; i +=
1) {
                        if (appInfo.fields[i].name ===
'dAppEndPoint') {
                            serverUrl =
appInfo.rows[appInfo.currentRow][i];
                        } else if (appInfo.fields[i].name ===
'dIsAppEnabled') {
                            enabled = appInfo.rows[appInfo.currentRow]
[i];
                        }
                        if (serverUrl && enabled) {
                            break;
                        }
                        if (enabled !== '1') {
                            serverUrl = '';
                        }
                    }
                    console.log('Get VBCS server from Idc Service: ' +
serverUrl);
                    resolve({'url': serverUrl});
                },
                error: function (xhr, status, err) {
                    console.log('Request failed: url:' + configUrl + '
status: ' + status + ' error: ' + err);
                    resolve({'url': serverUrl});
                }
            });
        });
    }
    return serverPromise;
};

```

5. Submit the request.

```

self.sendRequest = function (data, event) {
    var vbcsServer = self.VBCSServerUrl();
    var appName = 'requestform',
        appVersion = 'live',
        businessObject = 'Registration';

```

```

    var url = vbcsServer + '/rt/' + appName + '/' + appVersion + '/
resources/data/' + businessObject;
    var payload = {
        'firstName': self.firstName(),
        'lastName': self.lastName(),
        'email': self.email(),
        'phone': self.phone(),
        'company': self.company(),
        'jobTitle': self.jobTitle()
    };

    $.ajax({
        type: 'POST',
        url: url,
        beforeSend: function (xhr) {
            xhr.setRequestHeader('Content-type', 'application/
vnd.oracle.adf.resourceitem+json');
        },
        data: JSON.stringify(payload),
        dataType: 'json',
        success: function (data) {
            self.requestFailMsg('');
            self.requestSuccessMsg('Request has been submitted
successfully');
            self.firstName('');
            self.lastName('');
            self.email('');
            self.phone('');
            self.company('');
            self.jobTitle('');
            self.showDownload(true);
        },
        error: function (jqXHR, textStatus, errorThrown) {
            console.log('Error:');
            console.log(jqXHR);
            self.requestSuccessMsg('');
            self.requestFailMsg('Failed to submit the request');
        }
    });
};

```

6. Create a trigger to download a document.

```

{
    "id": "ECVBCS-Gated-Form",
    "settingsData": {
        "settingsHeight": 90,
        "settingsWidth": 300,
        "settingsRenderOption": "dialog",
        "componentLayouts": [],
        "triggers": [{
            "triggerName": "VBCSGatedFormSubmitted",
            "triggerDescription": "VBCS gated form submitted",
            "triggerPayload": [{

```

```

        "name": "payloadData",
        "displayName": "Document URL"
      }
    ]],
    "actions": []
  },
}

```

Register the trigger in appinfo.json.

```

self.raiseTrigger = function (triggerName) {
  SitesSDK.publish(SitesSDK.MESSAGE_TYPES.TRIGGER_ACTIONS, {
    'triggerName': triggerName,
    'triggerPayload': {
      'payloadData': 'https://docs.oracle.com/en/cloud/paas/content-
cloud/developer/developing-oracle-content-management-cloud.pdf'
    }
  });
};

self.startDownload = function (data, event) {
  console.log('Raise trigger: VBCSGatedFormSubmitted');
  self.raiseTrigger("VBCSGatedFormSubmitted"); // matches appinfo.json
};

```

- Raise the trigger in render.js.

\

Modify styles/design.css

Add the following CSS to design.css.

```

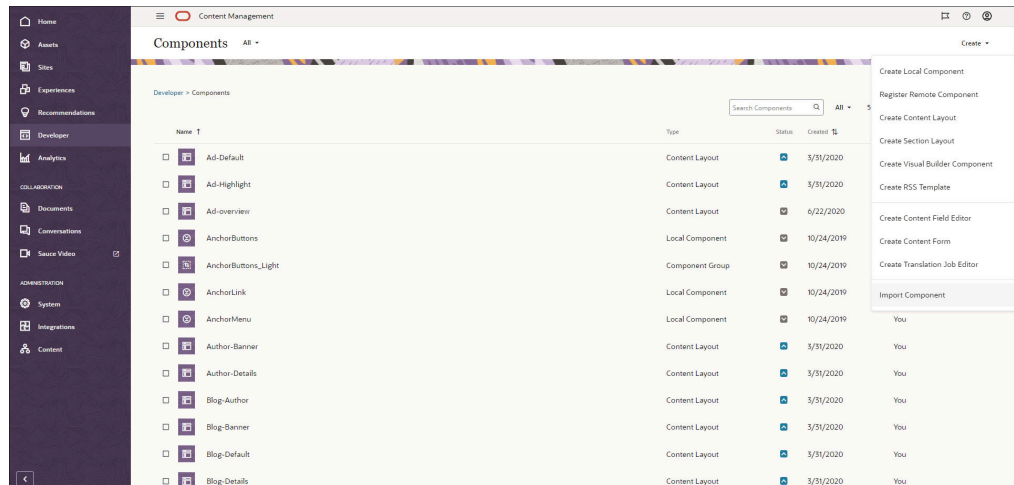
.form {
  font-family: "Helvetica Neue", "Segoe UI", sans-serif-regular,
Helvetica, Arial, sans-serif;
  font-size: 14px;
}
.form input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
.form textarea {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

```

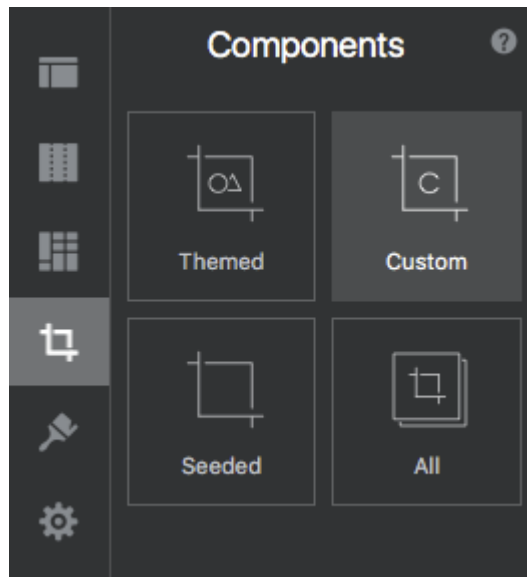
```
}  
.form button {  
  width: 100%;  
  background-color: #4CAF50;  
  color: white;  
  padding: 14px 20px;  
  margin: 8px 0;  
  border: none;  
  border-radius: 4px;  
  cursor: pointer;  
}  
.form button:hover {  
  background-color: #45a049;  
}  
.form button:disabled {  
  background-color: #dddddd;  
}  
.required-field::after {  
  content: "*";  
  color: red;  
  margin-left: 2px;  
}  
.request-msg {  
  padding: 5px;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  margin-bottom: 20px;  
}  
.green {  
  background-color: #81BA5E;  
}  
.red {  
  background-color: red;  
}
```

Use the Form Component on Oracle Content Management

1. Configure the VBCS connection:
 - Choose **Administration**, then **Integrations**, and then **Applications**.
 - Click the **Visual Builder Cloud Service Integration** check box.
 - Enter the URL, and click **Save**.
2. Import the component:
 - Choose **Developer** and then **Components**.
 - Choose **Create** and then **Import Component**.



3. Add the component to a page:
 - a. Edit a new or existing site.
 - b. In Site Builder, choose **Components** and then **Custom**.



- c. Drag the component onto the page.

The image shows a screenshot of a VBCS public form component. It contains the following elements:

- Name ***: A text input field with the placeholder text "Your name...."
- Email ***: A text input field with the placeholder text "Your email...."
- Phone**: A text input field.
- Subject**: A text input field.
- Message**: A large text area for entering a message.
- Send Request**: A grey button at the bottom of the form.



Note:

This VBCS public form component can be used on public or secure sites.

Build an Oracle Content and Experience VBCS Data Report Component

You can build a local Oracle Content Management component that uses REST APIs exposed by business objects in VBCS to deliver reports on data collected through forms.

Use data from a public form component to show the number of requests per day in a CSS bar chart. See [Build an Oracle Content Management VBCS Public Form Component](#).

Build an Oracle Content Management Local Component

Assumptions:

- The VBCS app name is "RequestForm".
- The business object name is "requestform".

Modify assets/render.js

1. Define the component template as follows.

```
<!-- ko if: initialized -->
<h1 style="text-align: center;">Number of requests per day</h1>
<div class="chartrow">
<div class="chartbody">
```



```
<div class="chartbody">
  <table id="q-graph">
    <tbody data-bind="foreach: requests">
      <tr class="qtr" data-bind="css: barcss">
        <td class="day bar" data-bind="style: {height:
height}">
          <p><span data-bind="text: value"></span></p>
        </td>
      </tr>
    </tbody>
  </table>

  <div id="ticks" data-bind="foreach: ticks">
    <div class="tick" style="height: 59px;">
      <p><span data-bind="text: value"></span></p>
    </div>
  </div>
</div>
<div class="chartlabel">
  <div class="labelrow">
    <div class="colorindex sunday"></div>
    <span>Sunday</span>
  </div>
  <div class="labelrow">
    <div class="colorindex monday"></div>
    <span>Monday</span>
  </div>
  <div class="labelrow">
    <div class="colorindex tuesday"></div>
    <span>Tuesday</span>
  </div>
  <div class="labelrow">
    <div class="colorindex wednesday"></div>
    <span>Wednesday</span>
  </div>
  <div class="labelrow">
    <div class="colorindex thursday"></div>
    <span>Thursday</span>
  </div>
  <div class="labelrow">
    <div class="colorindex friday"></div>
    <span>Friday</span>
  </div>
  <div class="labelrow">
    <div class="colorindex saturday"></div>
    <span>Saturday</span>
  </div>
</div>
</div>
<!-- note that the component has completed rendering into the page -->
<div class="scs-hidden" data-bind="scsRenderStatus: {'id': id, 'status':
'complete'}"></div>
<!-- /ko -->
```

2. Create the observables for the fields in the Knockout ViewModel.

```
self.initialized = ko.observable(false);
self.VBCSServerUrl = ko.observable();

self.requests = ko.observableArray();
self.ticks = ko.observableArray();

// Get VBCS server
var serverPromise = getVBCSServerURL();
serverPromise.then(function (result) {
    self.VBCSServerUrl(result.url);
    self.initialized(true);
    self.getRequests();
});
```

3. Obtain the VBCS connection.

See [Build an Oracle Content Management VBCS Public Form Component](#).

4. Get an Authorization Token.

Requirement: Oracle Content Management and VBCS are deployed in the same identity domain. See [Build an Oracle Content Management VBCS Secure Form Component](#).

5. Get requests.

Use a business object endpoint:

```
/Requestform

self.getRequests = function () {
    var vbcsServer = self.VBCSServerUrl();
    var authorization, token;
    var appName = 'requestform',
        mode = 'rt',
        appVersion = 'live'
        businessObject = 'Requestform';
    var url = vbcsServer + '/' + mode + '/' + appName + '/' +
appVersion + '/resources/data/' + businessObject;

    // get token first
    getAuthToken({
        'serverURL': self.VBCSServerUrl(),
        'successCallback': function (data) {
            token = data;
            authorization = (token.token_type ? token.token_type :
'Bearer') + ' ' + token.access_token;
            url = url + '?limit=999&orderBy=creationDate:desc';
            $.ajax({
                type: 'GET',
                url: url,
                beforeSend: function (xhr) {
                    xhr.setRequestHeader('Authorization',
authorization);
```

```

    },
    success: function (data) {
        if (data && data.count > 0) {
            self.showChart(data.items);
        }
    },
    error: function (jqXHR, textStatus, errorThrown) {
        console.log('Error:');
        console.log(jqXHR);
    }
});
});
},
'errorCallback': function (xhr, status, err) {
    if (xhr && xhr.status === 200) {
        token = xhr.responseText;
        console.log('Got token');
    } else {
        console.error('getToken: xhr: ' + JSON.stringify(xhr) + '
status: ' + status + ' error: ' + err);
    }
}
});
});
};

```

6. Set up chart data.

```

self.showChart = function (items) {
    var weekDayCounts = [0, 0, 0, 0, 0, 0, 0];
    var weekDayNames = ['Sunday', 'Monday', 'Tuesday', 'Wednesday',
'Thursday', 'Friday', 'Saturday'];
    for(var i = 0; i < items.length; i++) {
        var d = new Date(items[i].creationDate);
        var day = d.getDay();
        if (day >= 0 && day < 7) {
            weekDayCounts[day] = weekDayCounts[day] + 1;
        }
    }
    var max = 0;
    var results = [];
    for(var i = 0; i < weekDayCounts.length; i++) {
        if (max < weekDayCounts[i]) {
            max = weekDayCounts[i];
        }
    }
    var lines = 7,
    buckets = 5;
    var gap = Math.round(max / buckets);
    var top = gap * (lines - 1);
    if (max > top) {
        gap += 1;
    }
    var ticks = [];
    for(var i = 0; i < lines; i++) {
        ticks[i] = {
            value: i * gap

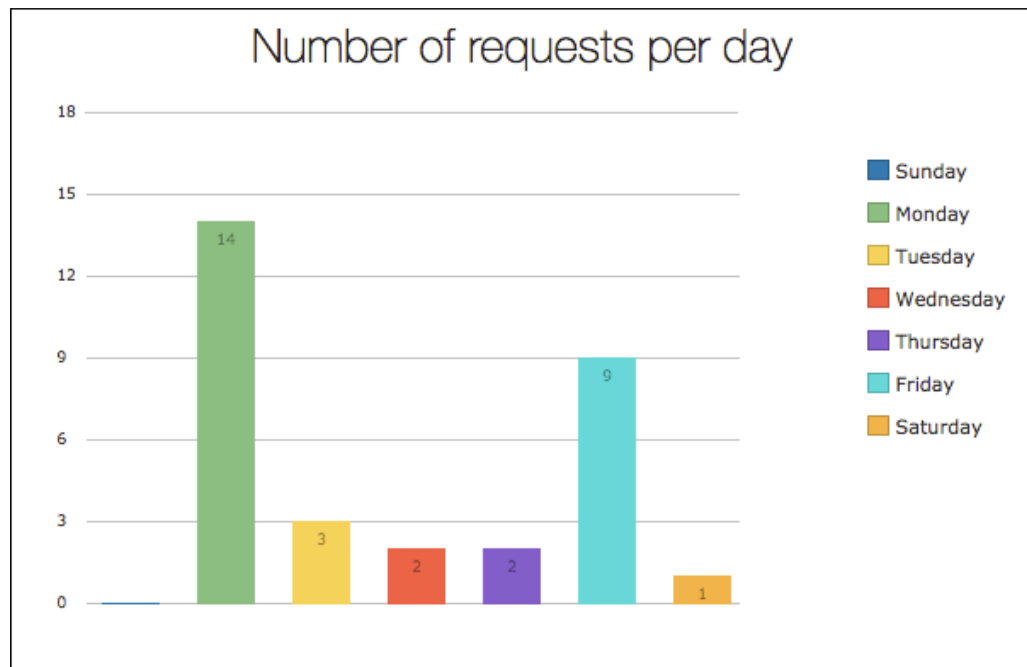
```

```
    }  
  }  
  self.ticks(ticks.reverse());  
  
  for(var i = 0; i < weekDayCounts.length; i++) {  
    var height = (weekDayCounts[i] / gap) * 60;  
    results.push({  
      height: height.toString() + 'px',  
      value: weekDayCounts[i],  
      barcss: weekDayNames[i].toLowerCase()  
    });  
  }  
  self.requests(results);  
};
```

Use the Component on Oracle Content Management

For information about how to add the component on a page, see [Build an Oracle Content Management VBCS Public Form Component](#).

The component renders as follows.



This VBCS component works only on secure sites.

Build an Oracle Content Management VBCS Multipage Form As a Web App Component

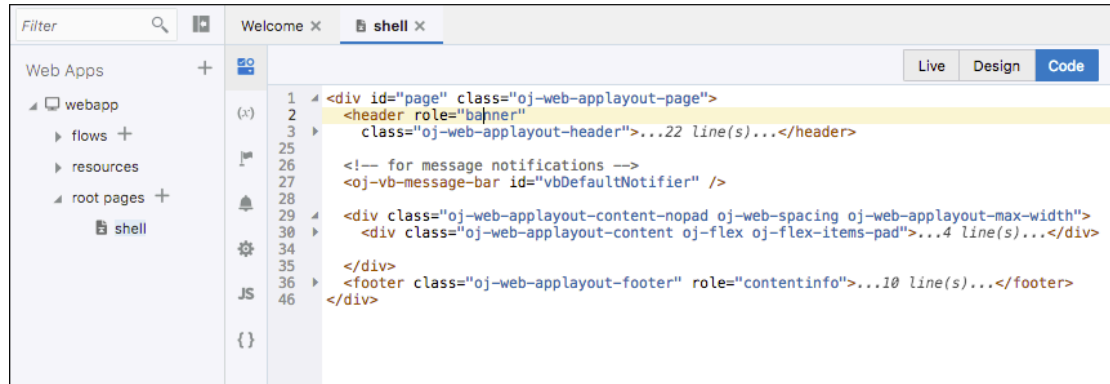
You can build a complex, multipage form for Oracle Content Management as a component of a VBCS web application.

To build a VBCS multipage form as a web application component for Oracle Content Management, you can remove the header and footer, set the IFrame height, add

validation for required form fields, .Call a REST endpoint to save form data, and, for a public web application, set security to allow anonymous access.

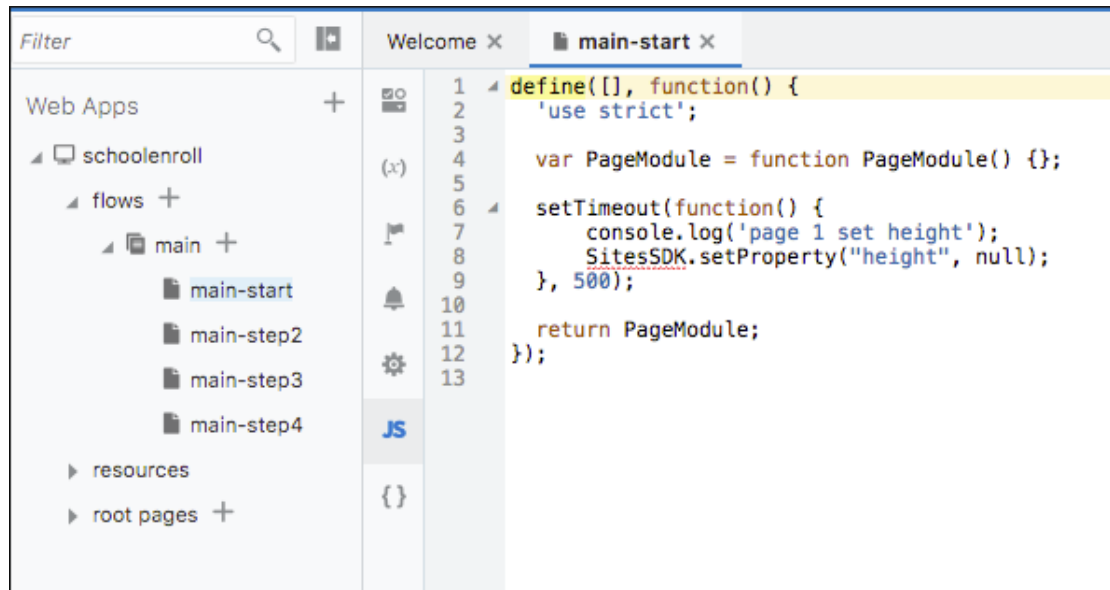
Remove the VBCS Web Application Header and Footer

If you don't want the default VBCS web application's header and footer, you can remove them from the page.



Set the VBCS Component Iframe Height

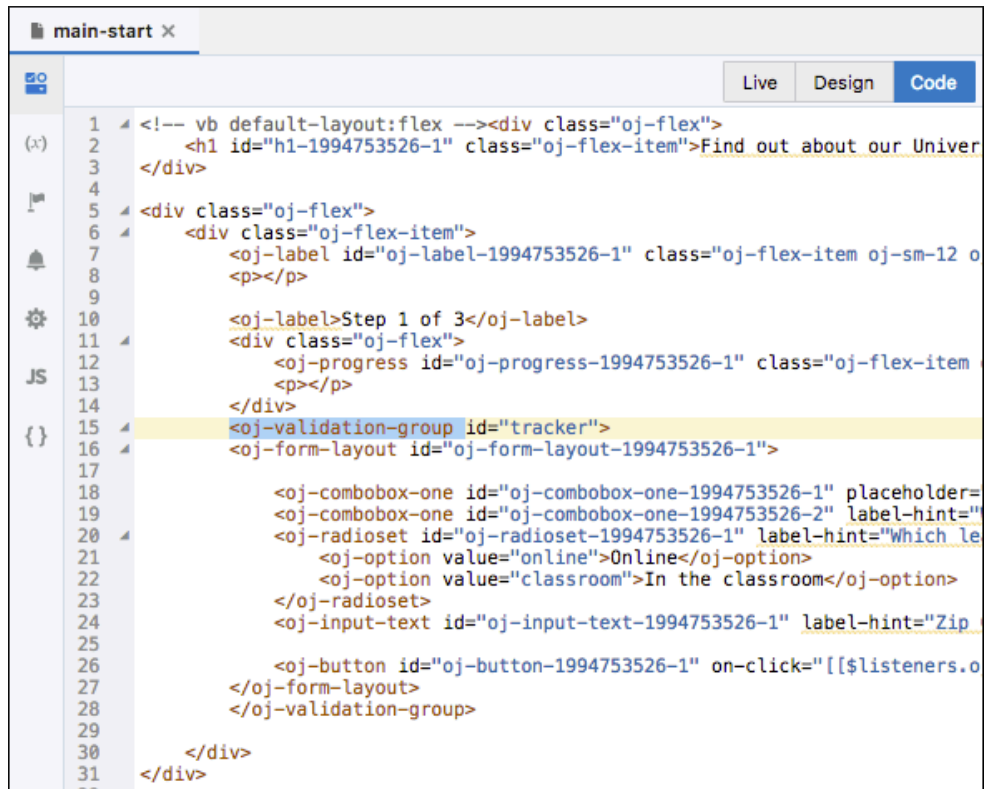
You need to set the VBCS component iframe height for each page in the flow of the web app. Include a validation group in the page model,



Required Form Fields Validation

When you try to navigate to another page, some of form fields on the current page are marked as "required" and don't have values yet. By default, there is no validation. You can use `oj-validation-group` to add the validation:

1. Wrap all your form fields inside the component `<oj-validation-group>`.



```
1 <!-- vb default-layout:flex --><div class="oj-flex">
2   <h1 id="h1-1994753526-1" class="oj-flex-item">Find out about our Univer
3 </div>
4
5 <div class="oj-flex">
6   <div class="oj-flex-item">
7     <oj-label id="oj-label-1994753526-1" class="oj-flex-item oj-sm-12 o
8     <p></p>
9
10    <oj-label>Step 1 of 3</oj-label>
11    <div class="oj-flex">
12      <oj-progress id="oj-progress-1994753526-1" class="oj-flex-item
13      <p></p>
14    </div>
15    <oj-validation-group id="tracker">
16      <oj-form-layout id="oj-form-layout-1994753526-1">
17
18        <oj-combobox-one id="oj-combobox-one-1994753526-1" placeholder=
19        <oj-combobox-one id="oj-combobox-one-1994753526-2" label-hint="
20        <oj-radioset id="oj-radioset-1994753526-1" label-hint="Which le
21          <oj-option value="online">Online</oj-option>
22          <oj-option value="classroom">In the classroom</oj-option>
23        </oj-radioset>
24        <oj-input-text id="oj-input-text-1994753526-1" label-hint="Zip
25
26        <oj-button id="oj-button-1994753526-1" on-click="[[listeners.o
27      </oj-form-layout>
28    </oj-validation-group>
29
30  </div>
31 </div>
```

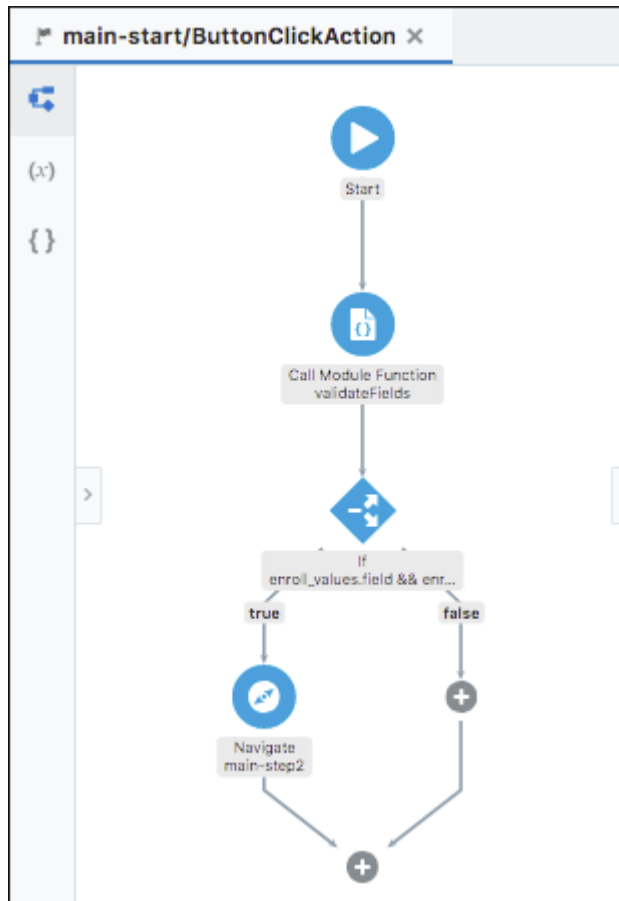
2. Include `oj-validation-group` in the page model.

```
main-start x
87 },
88 "imports": {
89   "components": {
90     "oj-label": {
91       "path": "ojs/ojlabel"
92     },
93     "oj-form-layout": {
94       "path": "ojs/ojformlayout"
95     },
96     "oj-input-text": {
97       "path": "ojs/ojinputtext"
98     },
99     "oj-combobox-one": {
100      "path": "ojs/ojselectcombobox"
101    },
102    "oj-radioset": {
103      "path": "ojs/ojradioset"
104    },
105    "oj-button": {
106      "path": "ojs/ojbutton"
107    },
108    "oj-progress": {
109      "path": "ojs/ojprogress"
110    },
111    "oj-validation-group": {
112      "path": "ojs/ojvalidationgroup"
113    }
114  }
115 }
```

3. Create a Flow module function .

```
main x
1 define([], function() {
2   'use strict';
3
4   var FlowModule = function FlowModule() {};
5
6   /**
7    *
8    * @param {String} arg1
9    * @return {String}
10   */
11   FlowModule.prototype.validateFields = function (arg1) {
12     console.log('validate fields:');
13     var tracker = document.getElementById("tracker");
14     if (!tracker) {
15       return;
16     }
17
18     tracker.showMessages();
19     if (tracker.valid === "invalidShown") {
20       tracker.focusOn("@firstInvalidShown");
21     }
22
23     setTimeout(function() {
24       console.log('validateFields set height');
25       SitesSDK.setProperty("height", null);
26     }, 500);
27
28   };
29
30   return FlowModule;
31 });
```

4. Create an action chain for the "Go to the Next Step" button.



The "If" condition checks the values for the required fields; for example:

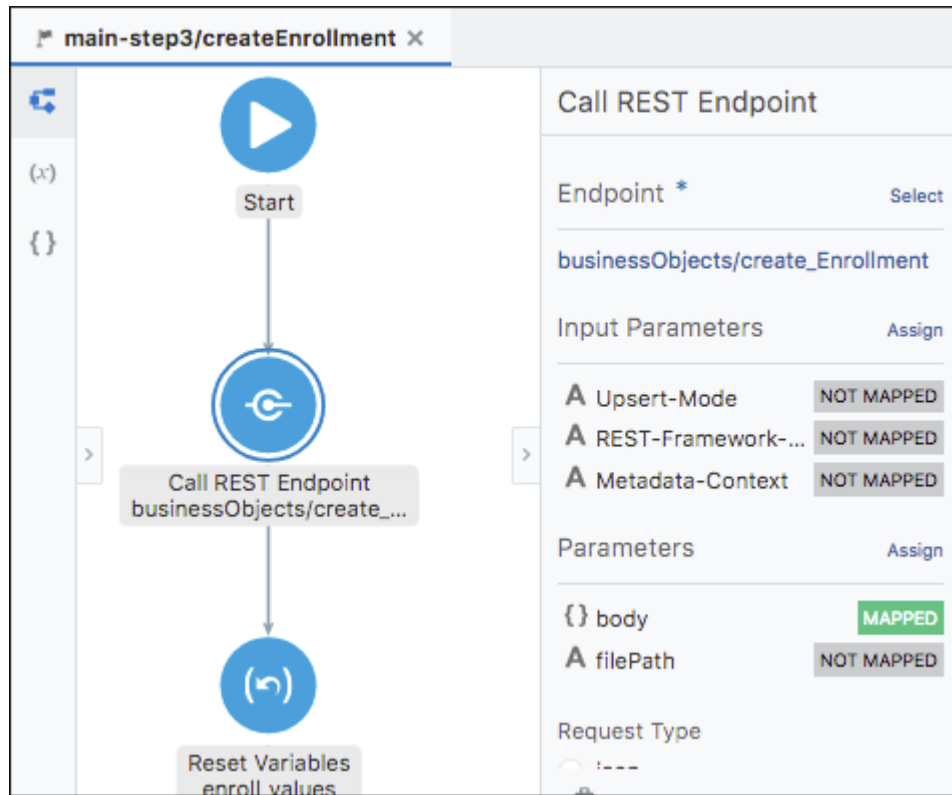
```

{{ $flow.variables.enroll_values.field
  && $flow.variables.enroll_values.zip }}
  
```

If the condition evaluates to true, navigation goes to the second step. Otherwise, it stays on the current page, and the error message is shown.

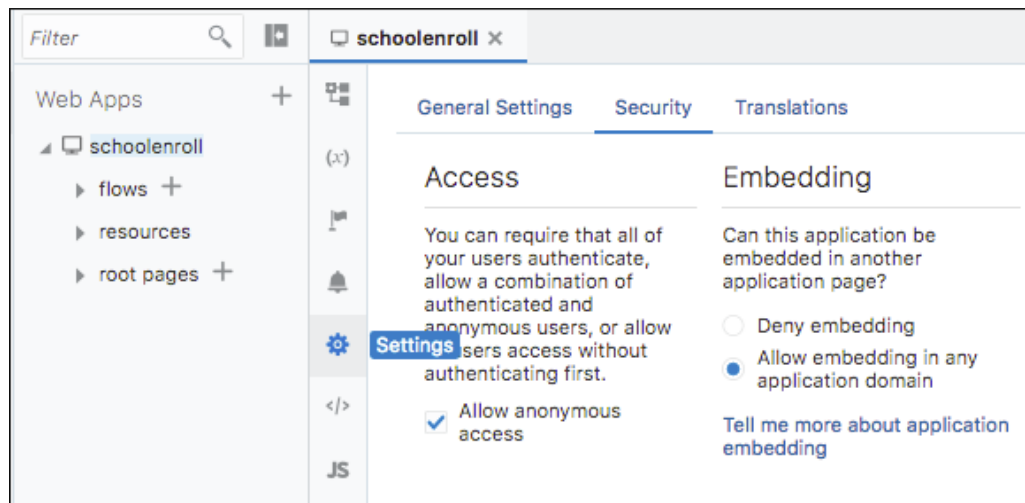
Call REST Endpoint to Save Form Data

At the last step, after the form fields validation, you can use the "Call REST Endpoint" action to persist form data in the business object.

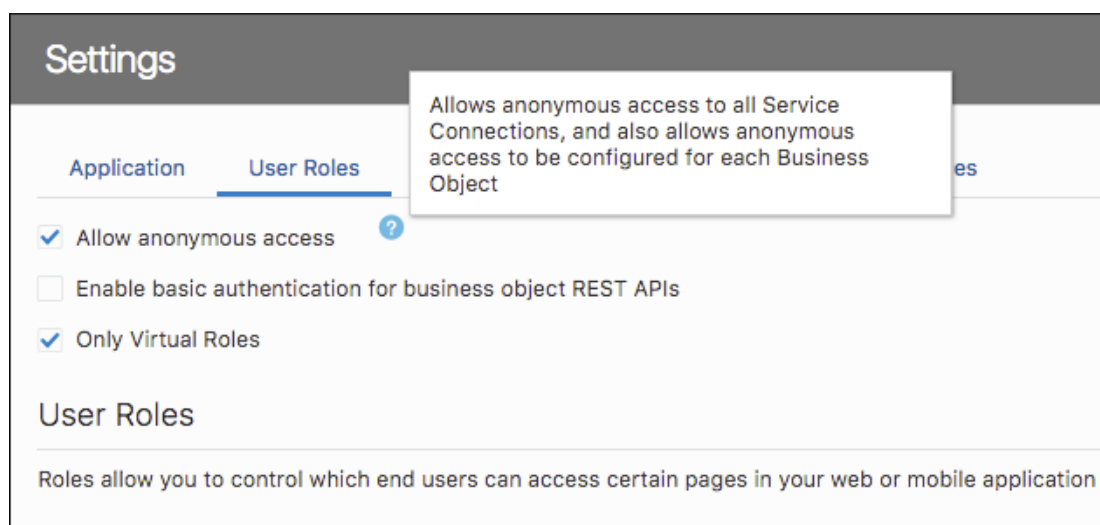


Public Web Application

If the web application is intended for public, you can set security to allow anonymous access.



Also allow anonymous access for the business object.



Provide a VBCS Endpoint As a URL for Select Menus

A content administrator can provide a Visual Builder Cloud Service (VBCS) endpoint as a URL to get values for select menus.

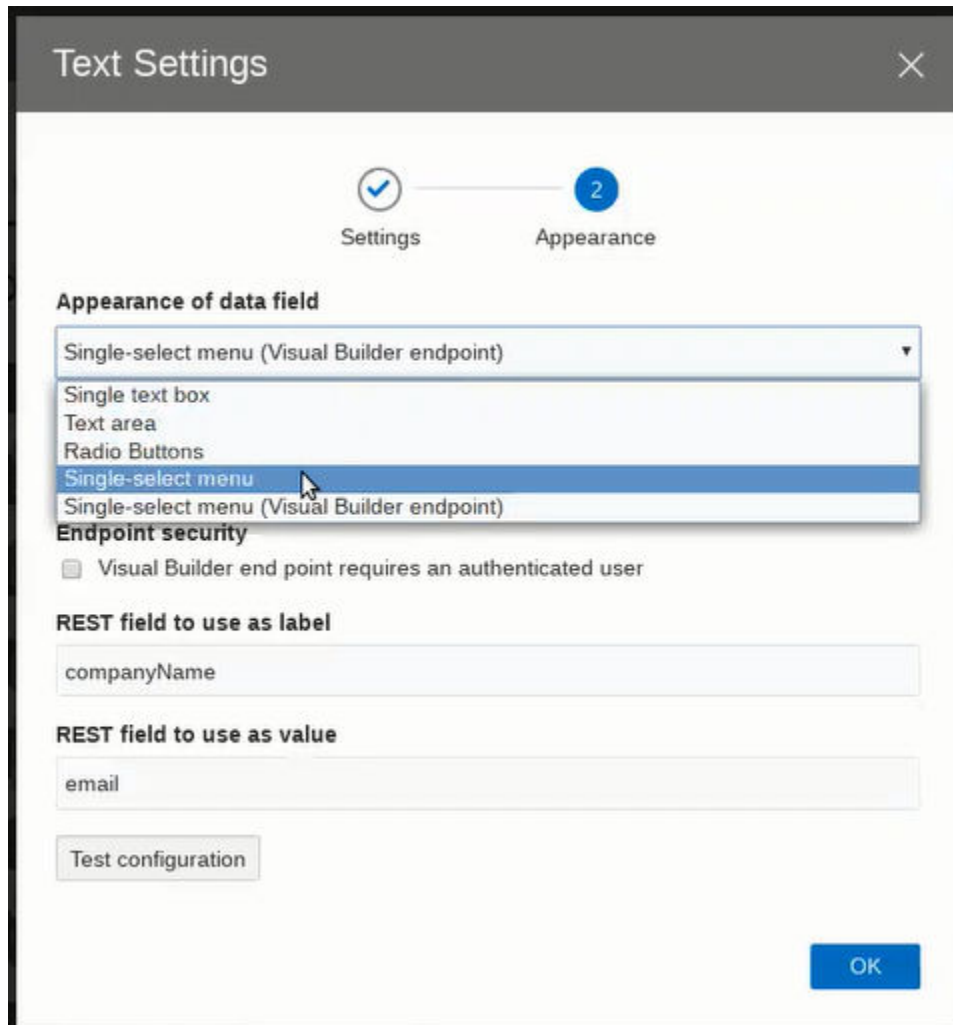
Instead of typing values for a select menu, users can enter a VBCS URL to get the values for the select menu in the Site Builder dialog. The content administrator can configure what attributes to use for option text and values in a REST response, using a public or secure VBCS URL.

Users can populate VBCS select menus in content item forms based on the URL configuration. When defining a content type and fields for the content type, the content administrator has the option to select a single-value menu that uses a Visual Builder endpoint.

The data for a content item is pooled by the Visual Builder URL. A business object, such as *Company*, can be defined by many fields in VBCS.

To create a content type with a VBCS endpoint as a URL for select menus:

1. Sign in as a content administrator in your browser and click **Assets** under Administration in the left navigation menu.
2. Choose **Content Types** from the drop-down menu and click **Create**.
3. Enter **VBCS** as the name of your content type.
4. Under **Content Type Definition** for the **VBCS** content type, click **Edit** to edit the settings.
5. In the **Text Settings** dialog, click **Appearance**.
6. Under **Appearance of data field**, choose **Single-select menu (Visual Builder endpoint)**.



7. In the **Visual Builder endpoint URL** field, enter a valid endpoint that is defined in VBCS.
8. For Endpoint security, select the check box if the Visual Builder endpoint requires an authenticated user. If the check box is not selected, anonymous users can use the VBCS endpoint.
9. Enter values for the **REST field to use as label** and **REST Field to use as value** fields, such as `companyName` and `email`.
10. Click **Test Configuration** to test the settings. If the configuration is correct, a "Test successful" message appears.
11. Save your changes and click **OK**.

Create Content Item

Content Type
VBCS

Name
Content item name
The item name should not contain the following characters: ~*?<%>[]/

Description (Optional)
Content item description

Collections
No collections to choose from

Language
English (United States) (en-US)
 Non-translatable

Targeted Channels
No channels to choose from

OK

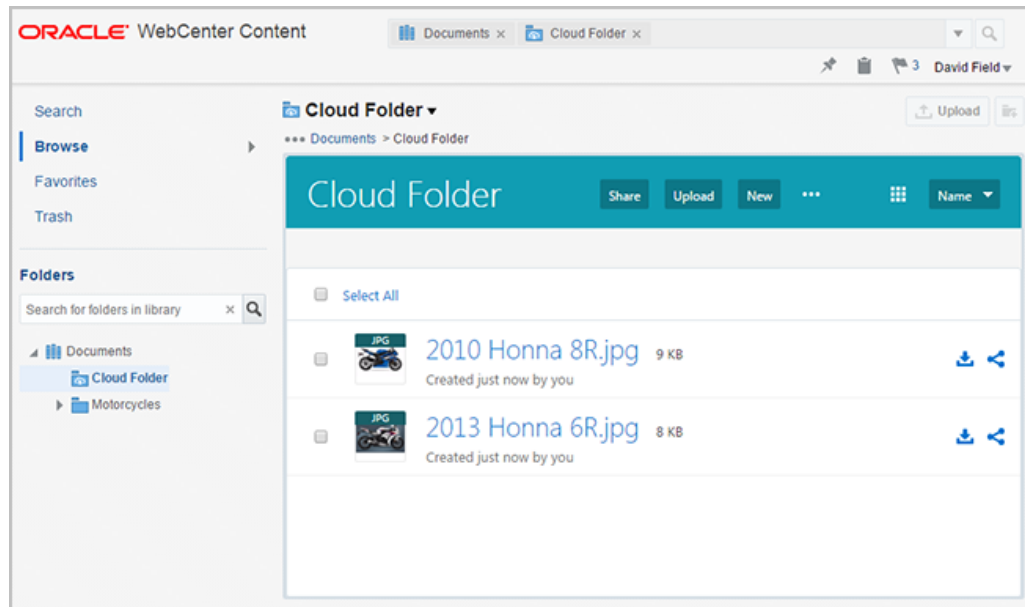
In the **Create Content Item** dialog, the UI for a VBCS content type looks identical to a single-select box for static data, except the data for the content item is dynamically pooled from the VBCS side with a run search. The value of **REST field to use as a value**, such as `email`, gets stored in the Oracle Content Management database.

Integrate with Oracle WebCenter Content

Why Integrate with Oracle WebCenter Content?

[Oracle WebCenter Content](#) belongs to [Oracle Fusion Middleware](#), a comprehensive family of software products that includes a range of application development tools and services. The [integration of Oracle Content Management with Oracle WebCenter Content](#) makes it easy to manage content in either environment and to share content between the two.

Need to collaborate with team members in the cloud? Users can [collaborate on cloud content](#) whether they are in the cloud, on their phone, or on-premise. This makes it possible for enterprises with investment in on-premise applications to leverage their investment while taking advantage of the rapidly expanding set of cloud offerings. Users can manage cloud content and move content between the cloud and on-premise installations.



When Oracle Content Management is enabled with Oracle WebCenter Content, you have a truly comprehensive hybrid enterprise content management (ECM) integration, with a unified ECM infrastructure and security from a single vendor. It combines anywhere access from the cloud with content retention and archiving from on-premise installations.

Prerequisites

There are prerequisites to integrating Oracle Content Management with Oracle WebCenter Content.

- On the Oracle Content Management side, you will need an Oracle Cloud Account, an OCM instance and be assigned with the right access roles.
- Check [Oracle WebCenter Content documentation](#) for prerequisites on the other side.

Integration Process

Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM. [Setting up the integration in Oracle WebCenter Content](#) is as easy as 1–2–3:

1. Enable the component.
2. Configure the server connection.
3. Configure the host name verifier.

3

Integrate with Third-Party Applications

You can integrate Oracle Content Management (OCM) with third-party applications including Microsoft Office for the web, Microsoft Outlook, Desygner, Kaltura video management, and Slack. To enable these integrations, log in to the Oracle Content Management web interface as an administrator, click **Integrations** in the Administration area of the navigation menu. On the Applications page, enable the third-party application integration that you'd want to make available in your instance. If you don't see this option, then you don't have the required privileges.

When integrating OCM with Microsoft Office applications such as Microsoft Word, PowerPoint, or Excel, the integration features become automatically available in these Office apps when users install the Oracle Content Management desktop app.



Note:

Additionally if you are an Oracle partner, you can create applications that use the integration features in Oracle Content Management. [Oracle Cloud Marketplace](#) is where you can publish these apps.

Explore the following topics:

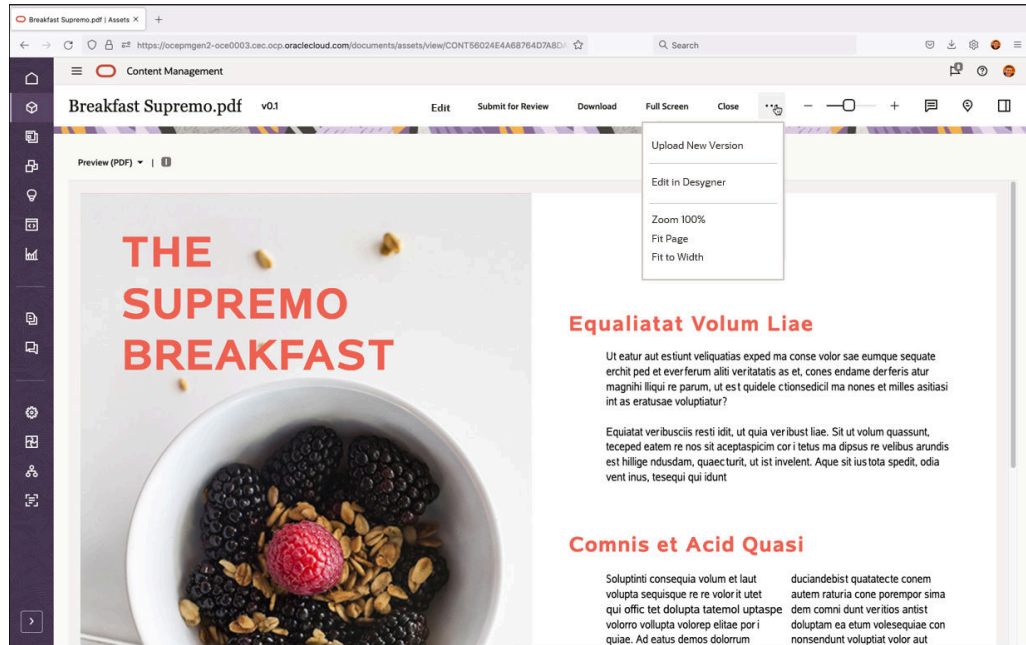
- [Integrate with Desygner](#)
- [Integrate with Kaltura Video Management](#)
- [Integrate with Microsoft Office](#)
- [Integrate with Slack](#)

Integrate with Desygner

Why Integrate with Desygner?

Desygner is a visual content creation solution for personal and business projects. It allows designers to create templates that other teams can use to create new assets while following template rules and layouts. As a result, organizations can efficiently create and edit print-ready documents that respect the organization's brand.

The integration of Desygner with Oracle Content Management (OCM) allows you (e.g., administrators, designers, content contributors) to take advantages of the features in both applications. For instance, you can create, edit, and manage Desygner assets from the Oracle Content Management web interface, leverage OCM as a cloud-based content management system to improve team collaboration, streamline content creation and publication, and promote content reuse from a single source (the OCM [repository](#)).



Prerequisites

There are prerequisites to integrating Desygnr with Oracle Content Management. On the Oracle Content Management side, you will need the following before the integration process:

- An Oracle Cloud Account.
- An Oracle Content Management instance.
- A subscription of [Desygnr](#).

Integration Process

Integrating Oracle Content Management with Desygnr consists of these steps:

1. [Add Desygnr as SAML Application in Oracle Identity Cloud Service](#).
2. [Create an Application User for Desygnr](#).
3. [Enable Desygnr Integration in Oracle Content Management](#).
4. [Work with Desygnr Asset Types](#).

Add Desygnr as SAML Application in Oracle Identity Cloud Service

This topic describes setting up Desygnr to use Oracle Identity Cloud Service (IDCS) for federated Single Sign-On (SSO). Although optional, federated SSO improves the user experience when Desygnr is integrated with Oracle Content Management, allowing seamless interaction between Oracle Content Management and Desygnr.

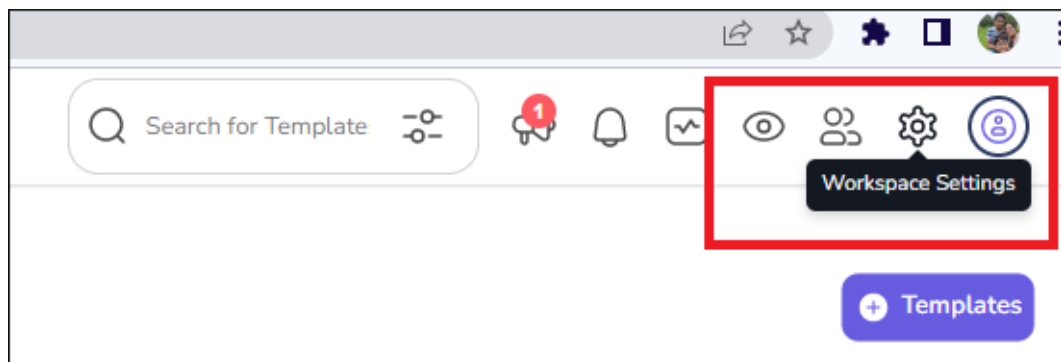
 **Note:**

Independent of the federated SSO setup, Desygnr needs an application user account to pass information from Desygnr to Oracle Content Management. The user account will be created as a special application user in IDCS.

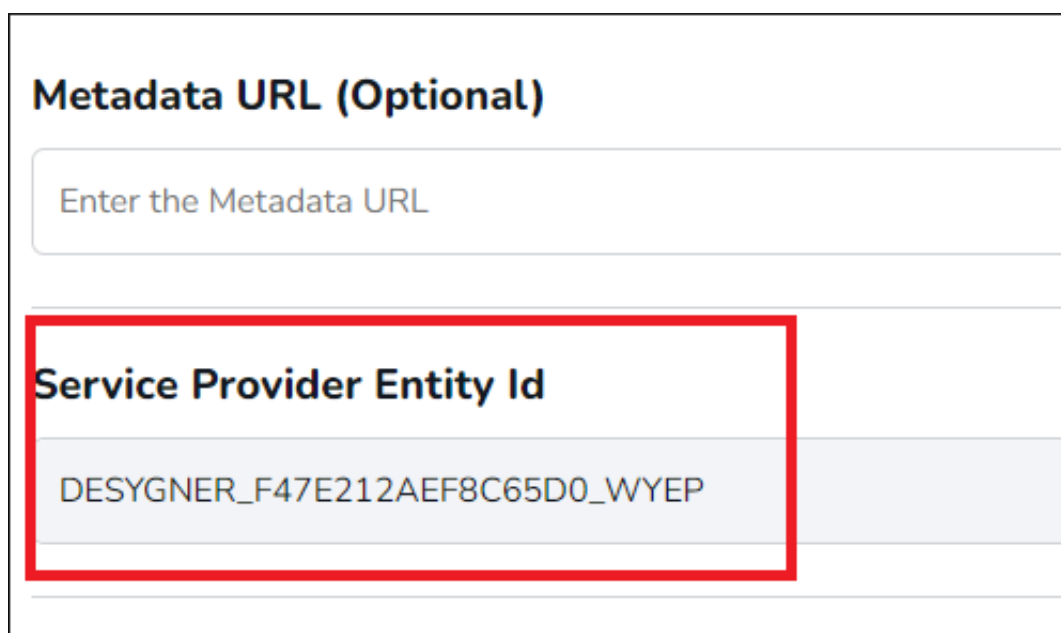
Obtain the Service Provider Entity ID from Desygnr

You obtain the Service Provider Entity ID from Desygnr and use it to configure the SAML application in IDCS.

1. Log in to the Desygnr application as an administrator. Click **Workspace Settings** in the top right corner. Click **Single Sign On** and then **Configure**.



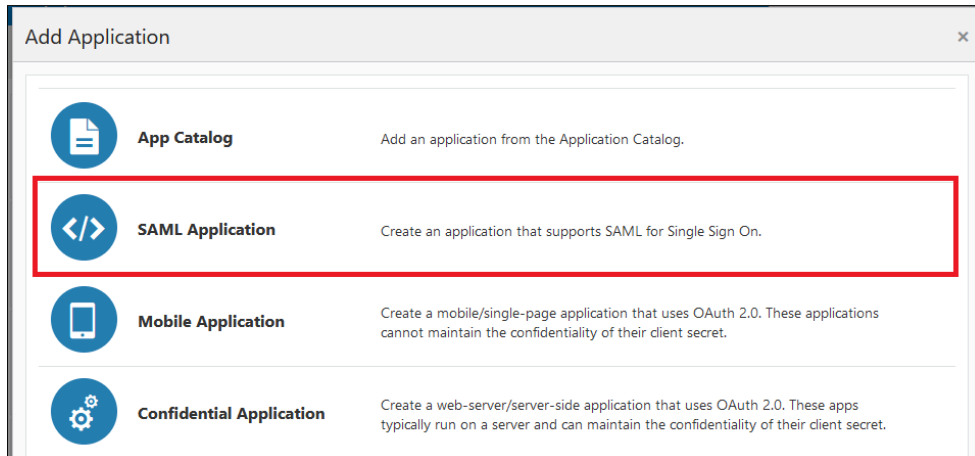
2. Note down the value of the Service Provider Entity ID. You will need this value, for example, *DESYGNER_F47E212AEF8C65D0_WYEP*, when configuring the SAML application in IDCS.



Create a SAML Application in Oracle Identity Cloud Service

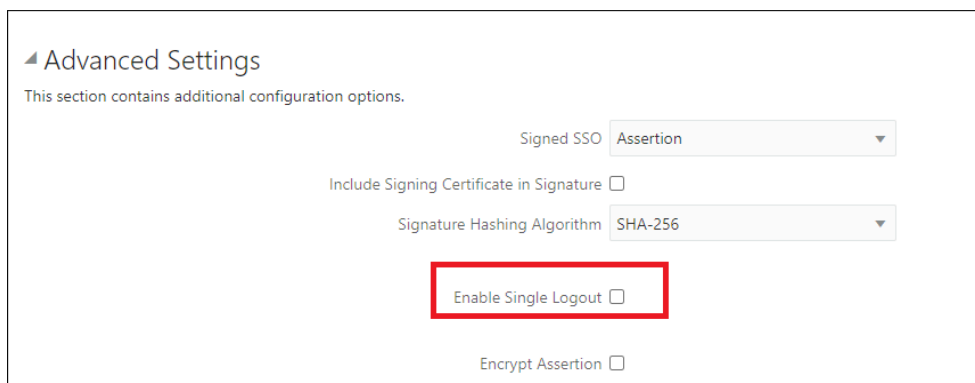
Create a Security Assertion Markup Language (SAML) application and grant it to users so that they can single sign-on (SSO) into the SaaS application that supports SAML for SSO.

1. Log in to the Oracle Identity Cloud Service (IDCS) admin console: <https://cloud.oracle.com/ui/v1/adminconsole>
2. Click **Applications and Services**. Click **Add** to open the dialog. Select **SAML Application**.



3. Enter a meaningful name for the application, for example, *Desygner Connector to OCM*. For the Application URL, enter `/connect/redirect`. Click **Next**.
4. In the **General** section, do the following:
 - a. Enter the value of the Entity ID which you noted from the Desygner Settings above, for example, `DESYGNER_F47E212AEF8C65D0_WYEP`
 - b. In the **Assertion Consumer URL** field, enter `https://<desygnerHost>/saml/acs`

Leave all the other fields with their default settings.
5. In the **Advanced Settings** section, uncheck the **Enable Single Logout** option.



- In the **Attribute Configuration** section, enter the attributes as shown in the screenshot below. The attributes map the users between IDCS and Desygnr.

Name	Format	Type	Value	Condition	Value
userName	Basic	User Attribute	User Name		

- In the **Authentication and Authorization** section, select the checkbox for **Enforce Grants as Authorization**, and click **Finish**.

Assign the Desygnr Role to Users in Oracle Identity Cloud Service

Oracle Identity Cloud Service (IDCS) manages user access to Oracle Content Management and Desygnr. It's recommended that you create an IDCS group with the necessary access roles and add users to the group to inherit the role. This example shows how individual users are assigned to Desygnr.

- Click the **Users** tab in IDCS.
- Click **Activate**, then **+Assign** to add users who will work with Desygnr from Oracle Content Management.
- Under **SSO Configuration**, click **Download Identify Provider Metadata** to download the `IDCSMetadata.xml` file to your local folder.

Configure Single Sign-On in Desygnr

Open the `IDCSMetadata.xml` file on your local machine to extract the information that's needed to configure Single Sign-On in Desygnr. The steps below describe how you would find the following fields in the XML file:

- IDP/SLO URL**
- EntityID**
- X509 Certificate**

- In the `IDCSMetadata.xml` file downloaded earlier, search for the **IDP/SLO URL**, the **EntityID**, and the **X509 Certificate** values, for example:

- `https://idcs-a9e0f119e3d04eef8287b584b10354.identity.oraclecloud.com/fed/v1/idp/slo`
- `<dsig:X509Certificate> MIIDXzCCAkegAwIBAgIGAWiWtp/ [...] </dsig:X509Certificate><dsig:X509IssuerSerial>`

2. Return to Desygnr, click **Workspace Settings**, then **Single Sign On**, and do the following:
 - a. For the **SAML 2.0 Endpoint (HTTP)** field, enter the IDCS **IDP/SLO URL** from the `IDCSMetadata.xml` file, for example:
`https://idcs-a9e0f119e3d04eef8287b584b10354.identity.oraclecloud.com/fed/v1/idp/slo`
 - b. For the **Identity Provider Issuer** field, enter the same IDCS URL but change the suffix to `/fed`. This will also be shown as the **EntityID** field in the `IDCSMetadata.xml` file, for example,
`https://idcs-a9e0f119e3d04eef8287b584b10354.identity.oraclecloud.com/fed`

Service Provider Entity Id

DESYGNER_F47E212AEF8C65D0_WYEP

Service Provider Metadata URL

`https://oracle.qcdesygnr.xyz/saml/metadata`

SAML 2.0 Endpoint (HTTP)

`https://idcs-a9e0f119e3d04eefad8287b584b10354.identity.oraclecloud.com/fed/v1/idp/slo`

Identity Provider Issuer

`https://idcs-a9e0f119e3d04eefad8287b584b10354.identity.oraclecloud.com/fed`

- c. Under the **Public Certificate** field, add the entire **X509 Certificate** value.
- d. Finally, set the **User Attribute Mappings** section.

User Attribute Mapping

Attribute name	Target property
userName	email (required)
userName	id (required)
userName	first_name (required)

Test the SSO Configuration

Log in to Oracle Content Management as the user who was added in the SAML application created for Desygnr. In another tab, access the Desygnr URL, e.g., `https://<DesygnrHost>/login/sso`.

If the SSO configuration is successful, you will be redirected to the landing page of Desygnr.

User Mappings in Desygnr

Depending on the user access privileges, Desygnr creates a different access level for each user. To illustrate the scenario, this section describes an example of two hypothetical users, John Smith and Robin Fisher.

1. John Smith is a user in Desygnr and Oracle Identity Cloud Service (IDCS). The `userId` links to an existing Desygnr `userId` when he logs in to Desygnr from Oracle Content Management. The `userId` has enterprise privileges.
2. Robin Fisher is a user only in IDCS, but not in Desygnr. When she logs in to Oracle Content Management through IDCS and accesses Desygnr, the `userId` for Robin Fisher is created in Desygnr. This `userId` has Pro+ privileges. To access Desygnr for the first time, she would need to click **Forgot Password** and then reset the password.

Create an Application User for Desygnr

When an asset is updated in the Desygnr application, the asset is sent to Oracle Content Management (OCM) using the Oracle Content Management REST API. To invoke the API, Desygnr needs an application user in OCM and an associated secure OAuth token. For this flow, the token is requested for the client application. Use this token to call Oracle Content Management.

Create a Confidential Application for Desygnr


You set up the OAuth grant type, client credentials, to access the resource.

1. Log in to the Oracle Identity Cloud Service Admin Console, <https://cloud.oracle.com/ui/v1/adminconsole>
2. Click **Confidential Application**. Give the application a name, and optionally, a description. For example, *DesygnrOAuthApp*. Click **Next**.
3. Choose to configure this client as a `Client`.
4. In the **Allowed Grant Types** section, select the checkbox for **Client Credentials**.
5. Scroll down to the **Token Issuance Policy** section. Under the **Resources** section, click **Add Scope** to give the application access to the required Oracle Content Management instance.
6. Click the right arrow to select the scope. The only scope required is the one ending in `urn:opc:cec:all`. Select the checkbox next to it and then click **Add**.
7. Click **Next** until the end.
8. Click **Finish**.
9. Note the **Client ID** and **Client Secret** values for retrieving a token later.
10. Check **Activate** and click **Save** to enable the application.

Assign the Oracle Content Management Role to the Application

You assign the Oracle Content Management repository administrator role and other necessary roles to the Desygnr application.

1. Log in to the IDCS Admin Console, <https://cloud.oracle.com/ui/v1/adminconsole>

2. Click **Oracle Cloud Services**.
3. Select the Oracle Content Management **service instance** for which you need privileges. This instance name would have been provided by the account administrator.
4. Click **Application Roles**. Navigate to the role **CECRepositoryAdministrator**. Click  in the right-hand corner. In the dropdown, click **Assign Applications**.
5. Search and find the application you created in the previous step, *DesygnerOAuthApp*. Select the application user, **DesygnerOAuthApp**, in this example.
6. Similarly add the application to **CECServiceAdministrator** and **CECSitesAdministrator** roles.
7. Click **Save**.


Set Up the Webhook URL in Desygner

When editing is completed in Desygner, documents in Desygner and Oracle Content Management need to be synchronized. The synchronization is achieved by setting up the webhook URL in Desygner to generate an OAuth access token for Desygner to invoke the Oracle Content Management (OCM) REST API, which then updates the Desygner documents in OCM.



Note:

As a prerequisite to generating the OAuth access token, [a confidential app must be created](#) in IDCS.

1. Log in to the Desygner UI (e.g., <https://oracle.desygner.com/login>) as an administrator.
2. Click  in the top right corner and select **Integrations**.
3. Enter the **Webhook URL** field.

The webhook URL is a configurable URL in the Desygner API. The URL contains the following fragments:

- Desygnr API OCM webhook endpoint: `/brand/companies/oracle/webhook`
- IDCS two-legged OAuth information
- The OCM instance identifier

The final webhook URL is shown as follows:

```
https://api.desygnr.com/brand/companies/oracle/webhook?
idcs_id=<...>&idcs_secret=<...>&idcs_scope=<...>&idcs_account=<...>&instance
=<...>
```

- The `idcs_account` can be obtained from the SSO redirection at a URL like this:

```
https://<your idcs_account>.identity.oraclecloud.com/ui/vi/signin
```

Input the value in the webhook URL *without* the prefix `idcs-`, (e.g., `b8078481XXX...XXXXbelaf018b`).

- The OCM instance (e.g., `finalcXXXX.....XXXXXontrol`) can be obtained from your Oracle Content Management instance at a URL like this:

```
https://<your OCM instance>.cec.ocp.oraclecloud.com/documents/home
```

- The Client ID `idcs_id` (e.g., `522a37XXXXXX...XXXXX201f`), Client Secret `idcs_secret` (e.g., `4d6cbf82-XXXX...XXXX393a4ff21e`), and Scope `idcs_scope` (e.g., `...XXXXXX`) can be obtained from IDCS. Note that these need to be [URL encoded](#).

Here is an example of a configured webhook URL:

```
https://api.qcdesygnr.xyz/brand/companies/oracle/webhook?
idcs_id=a7525f35a25344dea2baab7457d3a00e&idcs_secret=dc16ee62-1b96-4a51-
a37a-72e2ef56c77d&idcs_scope=https%3A%2F%2F2E364A57C2D741B0A72B21240A542CD0.
cec.ocp.oraclecloud.com%3A443%2Furn%3Aopc%3Acec%3Aall&idcs_account=a9e0f119e
3d04eefad8287b584b10354&instance=sandbox-oce0002
```

Enable Desygnr Integration in Oracle Content Management

After you've done the integration steps in Oracle Identity Cloud Service and Desygnr, you continue the integration process by configuring the Desygnr connectors in Oracle Content Management to enable the connection between the two applications and associate the Desygnr asset types to your repository.

Create a Publishing Channel and an Asset Repository


To create asset repositories and publishing channels, click on the **Content** option in the left navigation menu (under ADMINISTRATION). On the Content page, access **Publishing Channels** and **Repositories** in the Content dropdown menu.

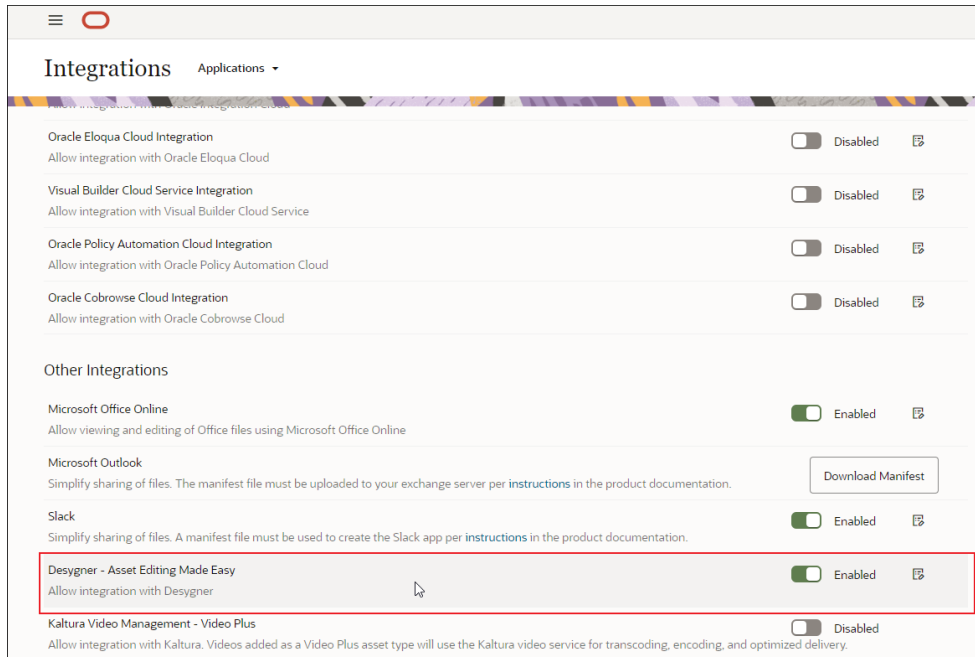
Note:

To set this up, you need to have the repository administrator role (CECRepositoryAdministrator) in Oracle Content Management.

1. Create a publishing channel.
2. Create an asset repository and associate it with the publishing channel that you created earlier.

Enable the Integration

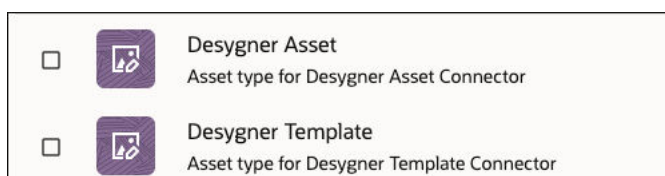
1. Log in to the Oracle Content Management web interface as an administrator.
2. Click **Integrations** in the left navigation menu and open the Applications page.
3. Enable the **Desygnr** option as seen here. Click  to set or change the Desygnr settings at any time.



4. The **Desygnr Configuration** dialog opens when you enable the integration. Configure the Desygnr connector settings. Input the connection information for Desygnr as needed. Some entries are pre-populated for you:

Field Name	Description
Description	A description of the integration.
Company Id	The <code>companyId</code> of your organization, which will be used in Desygnr REST API. For example, <code>oracle</code> .
REST URL	The REST URL for Desygnr, e.g., <code>https://api.desygnr.com</code>
Edit URL	The Edit URL for Desygnr that's used for editing the PDF documents, <code>https://<companyId>.desygnr.com/edit</code>

5. Once the connector configuration is completed, two connectors are enabled, Desygnr Asset and Desygnr Template. Additionally, two digital asset types of the same name are also created respectively. As an administrator, you can verify this by selecting **Content** (under ADMINISTRATION) in the left navigation menu of the OCM UI. Select **Asset Types** from the dropdown menu on the Content page. For details on asset types, see Manage Asset Types to learn more.



Enable the Integration on Asset Repositories

As a repository administrator, enable the Desygner integration on repositories of your choice to control who and how people access and use Desygner.

1. Associate the asset types with the repository of your choice. See Associate an Asset Type with a Repository for related details.

To create or edit the Asset Repository dialog, choose **Content** in the left navigation menu. On the Content page, choose **Repositories** in the Content dropdown menu. Select your repository to open **Edit Asset Repository**.

In the **Content Connectors** dropdown menu, Desygner Asset and Desygner Template connectors are available. Choose the connector based on your need. When **Desygner Asset** is selected, you can create a Desygner asset from a Desygner template. When **Desygner Template** is selected, you can import a Desygner template from Desygner to OCM. Automatically, the digital asset types are mapped to the repository.

Click **Save** when you're done.

2. The integration features immediately become visible in the repository, and you can perform tasks such as the following on the Assets page:
 - Import and edit the Desygner template.
 - Create a Desygner asset from the Desygner template.
 - Edit an asset in the Desygner UI.
 - Publish assets in the repository to the publishing channel.

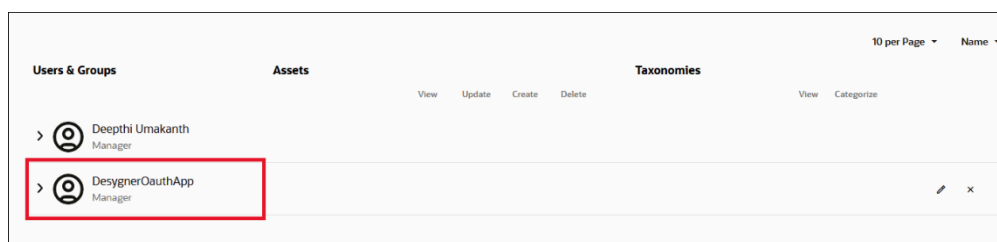
•  **Note:**

For details on the Assets page, see Get to Know the Assets Page.

3. You've completed the integration process. Share the repository with users so they can work with Desygner in Oracle Content Management.

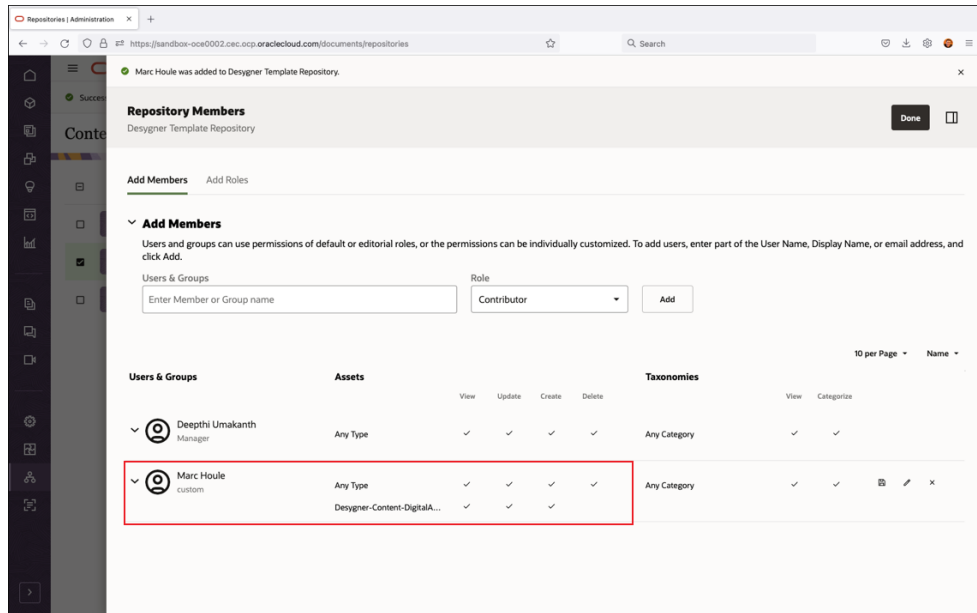
To add members and manage access rights, choose **Content** from the left navigation menu to open the **Repository** page. Select your repository and then click **Members** to access the **Repository Members** dialog.

Add the application user that you created earlier to the repository. Search for the application user's name (e.g., *DesygnerOAuthApp*) and add the user as a manager to the repository.



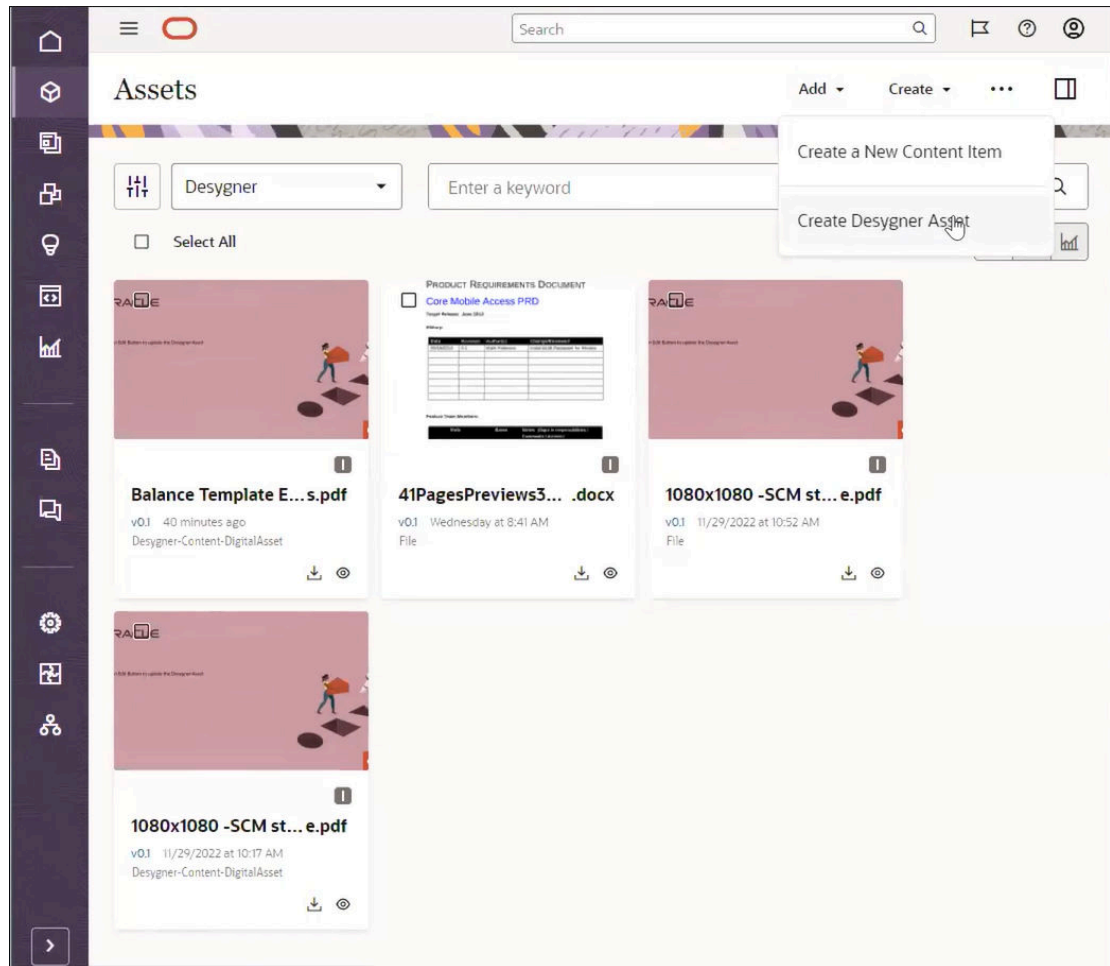
Additional users can be added at this time, and you can control their access of asset types. In the following screenshot, the highlighted user can view, create, and update Desygner assets in the repository, but cannot delete any.

See [Share a Repository and Manage Digital Assets](#) for related details.



Work with Desygnr Asset Types

After the integration with Desygnr is enabled, you can work with Desygnr asset types on the Assets page: import a template from Desygnr (under the **Add** dropdown menu), create, edit, and publish your Desygnr assets.



Integrate with Kaltura Video Management

Why Integrate with Kaltura Video Management?

The integration of Oracle Content Management with Kaltura allows videos that are added as a Video Plus asset type to use the Kaltura video service. Kaltura provides a rich video management and delivery experience, including all the standard features, such as upload, manage, preview, and download, plus advanced capabilities for optimized editing, streaming, automatic transcoding and conversion, and more responsive playback options.

Prerequisites

There are prerequisites to integrating Oracle Content Management with Kaltura Video Management - Video Plus. On the Oracle Content Management (OCM) side, you will need an [Oracle Cloud account](#), an [OCM instance](#), and be assigned with the right access role. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM.

Integration Process

To integrate OCM with Kaltura Video Management - Video Plus, log in to the Oracle Content Management web interface as an administrator, go to **Integrations** in the navigation menu on the left to open the Applications page. Enable **Kaltura Video Management - Video Plus**. If you don't see this option, then you don't have the required privileges.

Integrate with Microsoft Office

Why Integrate with Microsoft Office?

The integration of Oracle Content Management with Microsoft Office allows you to take advantages of features from both applications. Oracle Content Management can integrate with Microsoft Office in various ways:

- [Add Oracle Content as a save/open location in Microsoft Office applications.](#)
- [Access Oracle Content features within Microsoft Office applications.](#)
- [Add links to cloud items directly from Microsoft Outlook.](#)
- [Create or edit Microsoft Office files from the Oracle Content Management web interface.](#)



Note:

The supported versions for these integrations are Microsoft Office 2016, 2019, and 2021.

Prerequisites

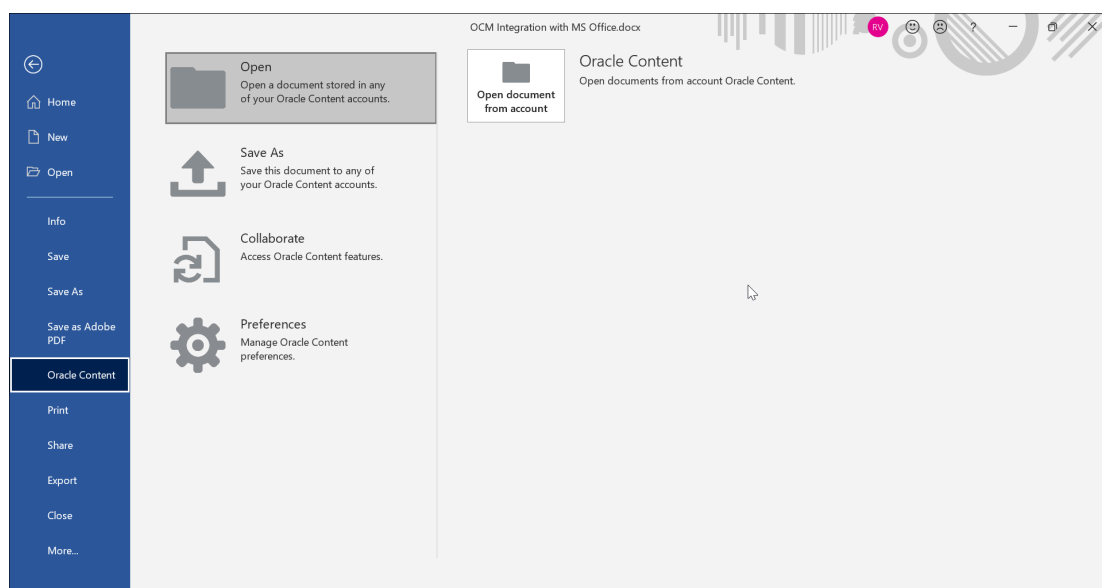
There are prerequisites to integrating Oracle Content Management with Microsoft Office. On the Oracle Content Management (OCM) side, you will need an [Oracle Cloud account](#), an [OCM instance](#), and be assigned with the right access role. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides. In addition to the requirements on the OCM side, you need a subscription for the product that you intend to integrate with OCM.

Integration Process

To integrate Oracle Content Management with Microsoft Office for the web and Microsoft Outlook, enable these integrations from the OCM web interface as an administrator. Click **Integrations** in the navigation menu on the left to display the Applications page. When integrating OCM with Microsoft Office applications such as Microsoft Word, PowerPoint, or Excel, the integration features become automatically available in these Office apps when users install the Oracle Content Management desktop app.

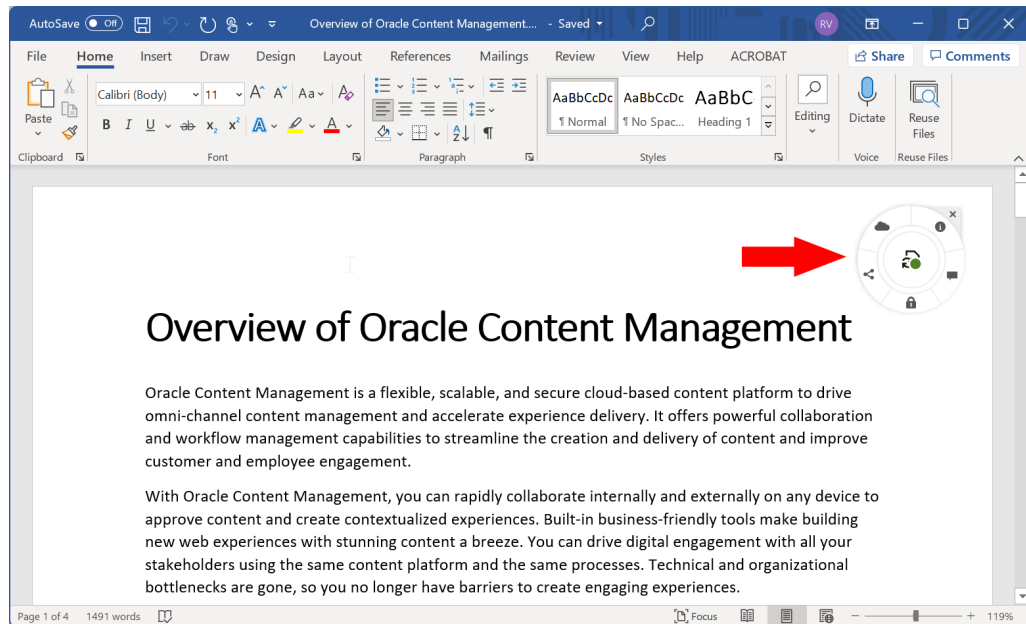
Add Oracle Content as a Save/Open Location in Microsoft Office Applications

When users install the Oracle Content Management desktop app, Oracle Content is automatically added as a location in Microsoft Word, Excel, and PowerPoint. This allows them to open documents directly from the cloud or save documents to the cloud. In addition, they can access some Oracle Content features for the current document, including sharing the document, starting or viewing a conversation about the document, or viewing its properties.



Access Oracle Content Features Within Microsoft Office Applications

When users install the Oracle Content Management desktop app, a special radial menu is automatically installed for Microsoft Word, PowerPoint, and Excel, which allows them to access Oracle Content features for the current document directly within these applications. The radial menu is displayed for all documents that are managed in Oracle Content Management.



Users can perform these tasks in the radial menu:

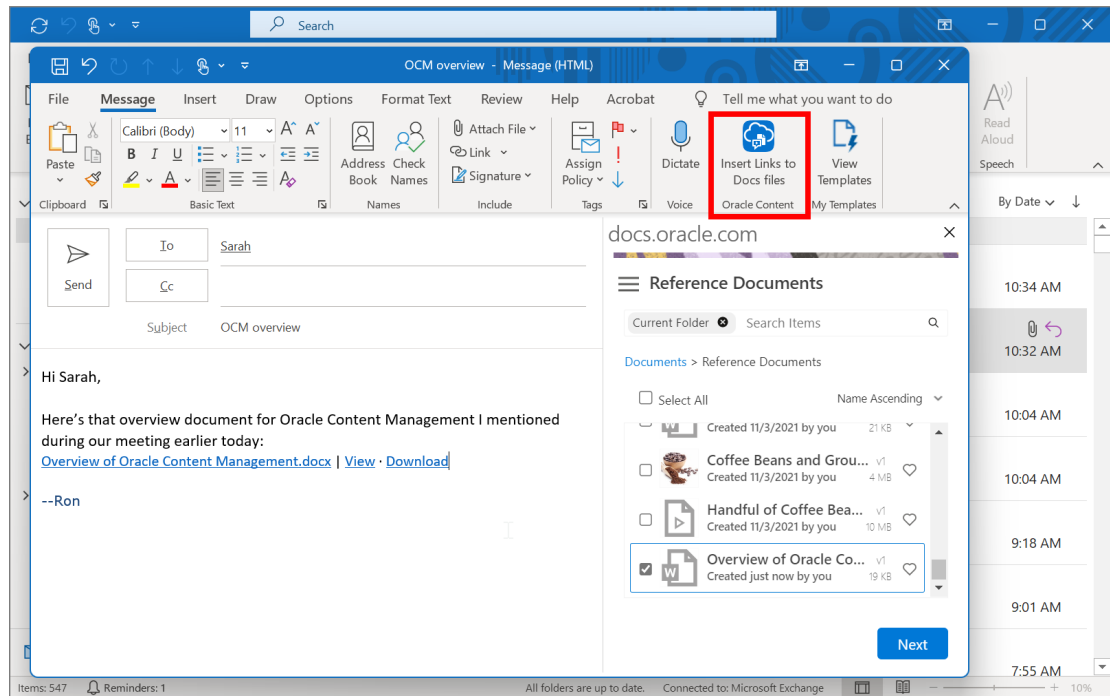
- Share the current document with other people.
- Lock the current document for editing, so other users can't inadvertently modify a document that you're working on.
- Start or view a conversation on the current document.
- View properties, access history, or version history of the current document.
- Go to the current document in the Oracle Content Management web interface.

Users can turn the radial menu off, either temporarily or permanently:

- Temporarily: click 'x' in the radial menu. The radial menu will reappear the next time a cloud document is opened.
- Permanently: open the **File** menu in the Microsoft Office application, choose **Oracle Content**, then **Preferences**, and disable the **Show Oracle Content radial menu when working with Microsoft Office documents** option. The radial menu will no longer appear when the user opens documents managed in Oracle Content Management.

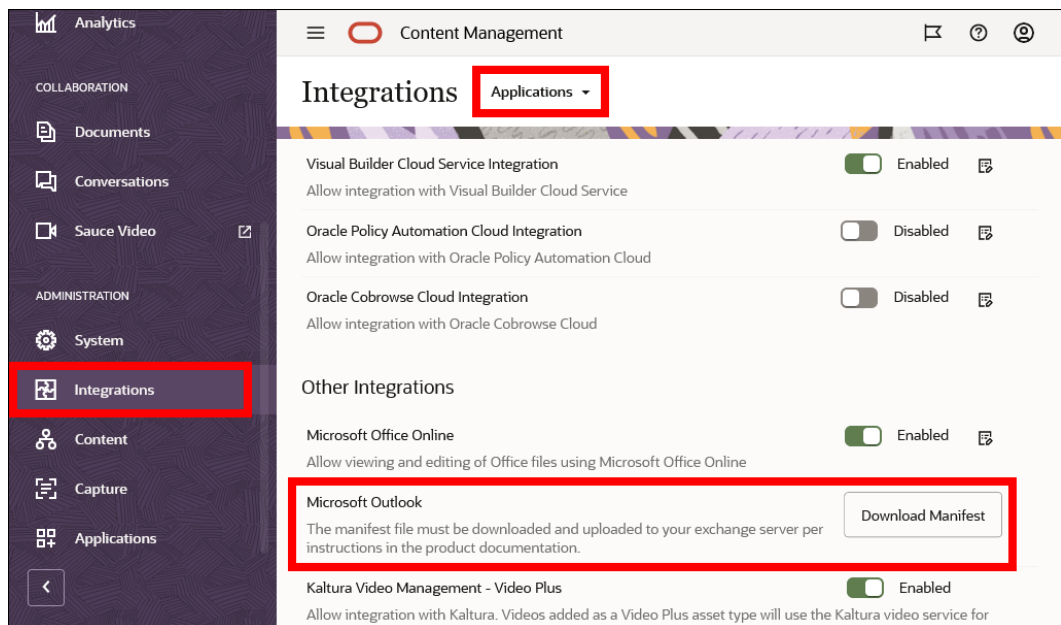
Add Links to Cloud Items Directly from Microsoft Outlook

Email administrators can install and configure Oracle Content Management for Outlook on a Microsoft Exchange server, so users can easily embed links to their Oracle Content files and folders into email messages or calendar appointments.



To set this up, deploy Oracle Content Management for Outlook on the Microsoft Exchange server:

1. Log in to the Oracle Content Management web interface as an administrator.
2. Click **Integrations** in the left navigation menu and open the Applications page.
3. Next to Microsoft Outlook, click **Download Manifest**.

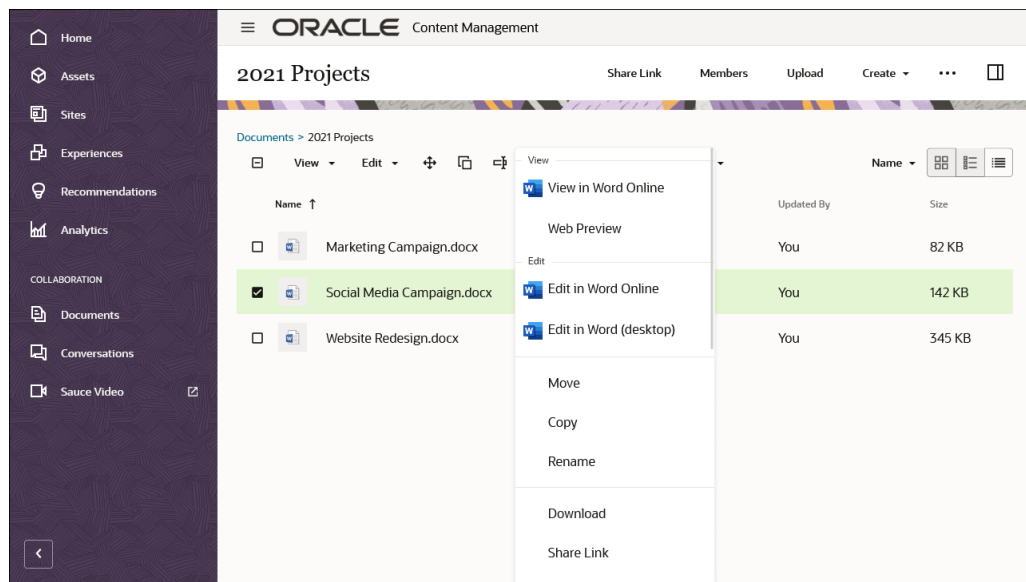


4. Install the add-in on your Microsoft Exchange server, following [Microsoft's documentation](#).
5. It can take up to 72 hours for the new add-in deployment to be available in users' Outlook applications. After you confirm that the add-in is visible to users:

- Inform them of the new Oracle Content Management add-in features.
- If your users previously installed the Oracle Content add-in for Outlook by installing the Oracle Content Management desktop app, they should disable that add-in.

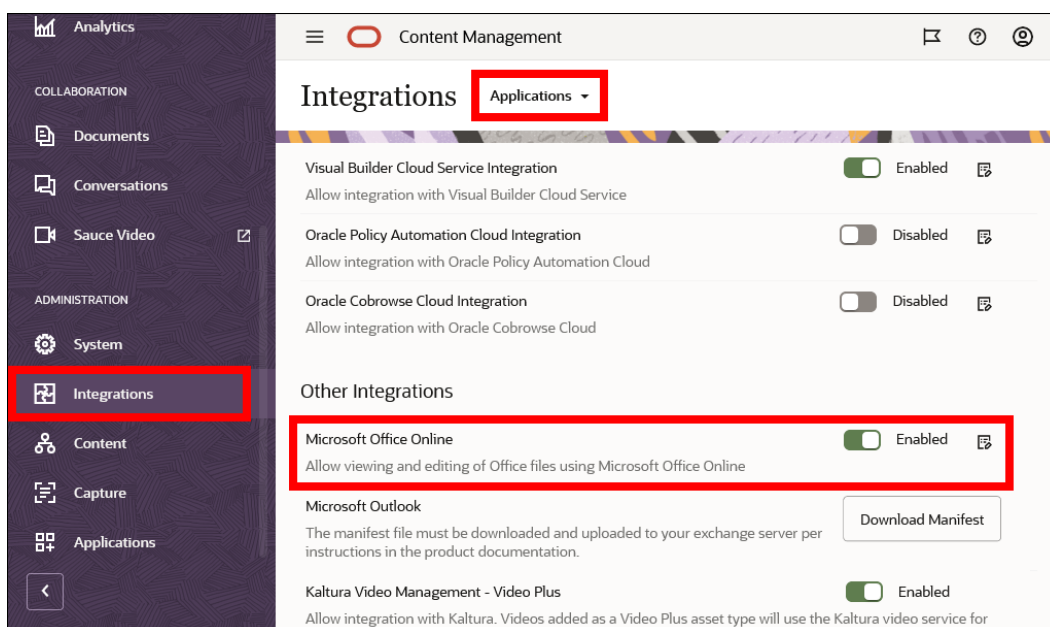
Create or Edit Microsoft Office Files from the Oracle Content Management Web Interface

Oracle Content Management can be configured to allow users to create, view, or edit Microsoft Office files in the Oracle Content Management web interface using the desktop Microsoft Office applications or Microsoft Office for the web (office.com).



To set this up, enable the Microsoft Office for the web integration for Oracle Content Management:

1. Log in to the Oracle Content Management web interface as an administrator.
2. Click **Integrations** in the left navigation menu and open the Applications page.
3. Enable the **Microsoft Office Online** option.



Note:

Optionally, you can include additional notification text that will appear in the pop-up window when a user opens an Office file from Oracle Content Management. Click



Integrate with Slack

Why Integrate with Slack?

The integration of Oracle Content Management with Slack enables you to create an app for Slack that is associated with an Oracle Content Management instance. This allows you to use Slack to share and collaborate on documents stored in Oracle Content Management and still control whether a Slack user has permissions to view, download, or edit shared documents.

Prerequisites

There are prerequisites to integrating Oracle Content Management with Slack. Integrations generally require configurations in both applications that are being integrated. So check the documentation on both sides.

- You must have the necessary accounts and permissions to create apps for Slack. The app for Slack is used when integrating with Oracle Content Management, and creating the app is done on the Slack website at <https://api.slack.com/apps>.
- On the Oracle Content Management (OCM) side, you will need an [Oracle Cloud account](#), an [OCM instance](#), and be assigned with the right access role.

Integration Process

Integrating Slack with Oracle Content Management consists of these steps:

1. [Enable and Configure Integration with Slack in Oracle Content Management](#). You must configure the integration for a specific Oracle Content Management instance.
2. [Create an App for Slack Using the Slack Website and Install](#). You'll need to create and install the app either to a specific workspace or as an enterprise [org app](#), and paste in the JSON-formatted configuration copied from the manifest when configuring the integration. Then by installing the app, the configuration is verified, and app credentials are generated.
3. [Add the App Credentials to Oracle Content Management](#). When the Slack app is installed, authentication credentials and tokens are generated which must be entered in the Oracle Content Management integration configuration dialog.

Enable and Configure Integration with Slack in Oracle Content Management

Integration with Slack is disabled by default. You must associate a specific Oracle Content Management instance with the app for Slack and provide app credentials so that Oracle Content Management has access to the Slack API.

1. Log in to the Oracle Content Management web interface as a service administrator.
2. Click **Integrations** in the Administration section of the navigation menu on the left. If you don't see this option, then you don't have the required access privileges.
3. Ensure integration with Slack is enabled. If the integration configuration dialog does not display, click the configuration icon.
4. In the first page of the integration with Slack configuration dialog, enter a name, description, and command name for the app for Slack. The command name is a slash /command shortcut you create to request access to your Oracle Content Management instance; for example, /sharecontent. Choose the command name carefully as it is difficult to change once it is defined.
5. Check **Enable as Enterprise Org app** if you are using an [Enterprise Grid](#).

Note:

If enabled as an enterprise org app, Slack users will still need to approve the app and select a workspace on which it will be used. This offers more control, as your organization may have hidden workspaces or workspaces with more stringent security requirements.

6. Click **Get Manifest**.

The manifest is a JSON-formatted configuration that includes the name and description you entered and the necessary URLs for Slack to handle requests, events, and authentication redirects associated with this specific Oracle Content Management instance.

7. Select and copy the text displayed in the manifest, then click **Save**.

 **Note:**

Don't exit out of the integration with Slack configuration dialog until after you've provided the client ID, the client secret, the signing secret, and the OAuth token. These app credentials are not generated until you've created the Slack app on the Slack website and installed the app.

Create an App for Slack Using the Slack Website and Install

Creating an app for Slack that is used when integrating with Oracle Content Management is done on the Slack website at <https://api.slack.com/apps>. You must have the necessary accounts and permissions to create Slack apps.

1. Make sure you are signed in to Slack and go to <https://api.slack.com/apps>.
2. Click **Create New App**.
3. Opt to create the app from an app manifest.
4. Select to install as an org app or select a workspace to create your app in. This is the workspace you'll install the app in as well. Once selected, click **Next**.
5. Overwrite the sample JSON-formatted manifest with the manifest you copied when configuring Oracle Content Management integration with Slack and click **Next**.
6. Review the summary. If anything is incorrect, you'll need to go back and manually edit the manifest.
7. Add the Oracle Content Management icon:
 - a. Download the icon from the Oracle content delivery network at <https://static.ocecdn.oraclecloud.com/cdn/cec/ocmslack/v1.0.0/images/ocm-1024.png>.
 - b. Scroll the summary page to the display information section and click **Add app icon**.
 - c. Browse to the Oracle Content Management icon and select it.
 - d. Set the background color to #564162 to match the color used in the icon.
8. When finished reviewing the summary and adding the app icon, click **Create**.
9. If installing the Slack app as an org app, you may need to check **Enable as Enterprise Org app**.
10. Click the install button to install the app and generate the OAuth token.

Add the App Credentials to Oracle Content Management

Once all the credentials required by Oracle Content Management are generated, they must be added to the configuration.

1. Copy the values for the client ID, the client secret, the signing secret, and the bot user OAuth token from the app for Slack information page to the appropriate fields on the second page of the integration with Slack configuration dialog in Oracle Content Management.
2. When finished, click **Save**.

3. Optionally, enable **Show content previews within Slack**. This option allows links used in Slack to show a preview image of content in Oracle Content Management, provided the Slack user has permissions to view the content.



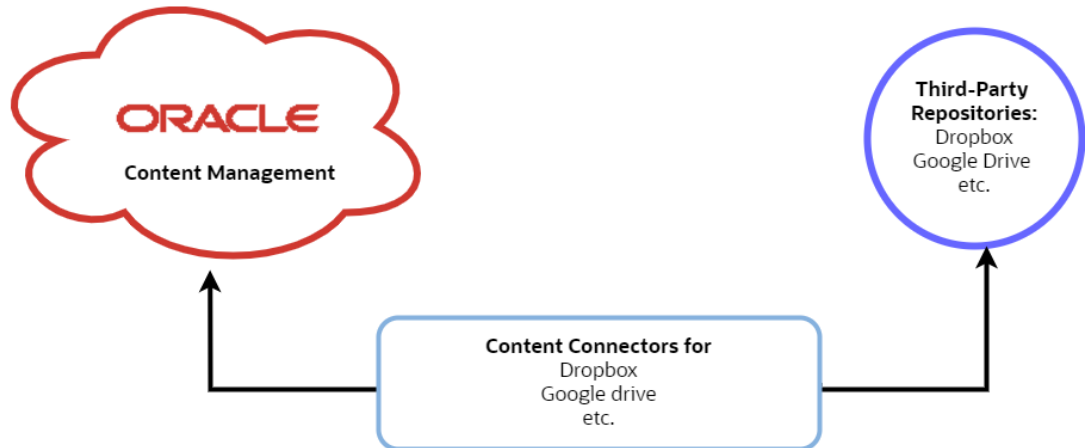
Note:

If installing the app for Slack to a specific workspace and the configuration is finished, everyone who needs access to the app will have to connect it to their Slack client using the Slack app directory in Slack.

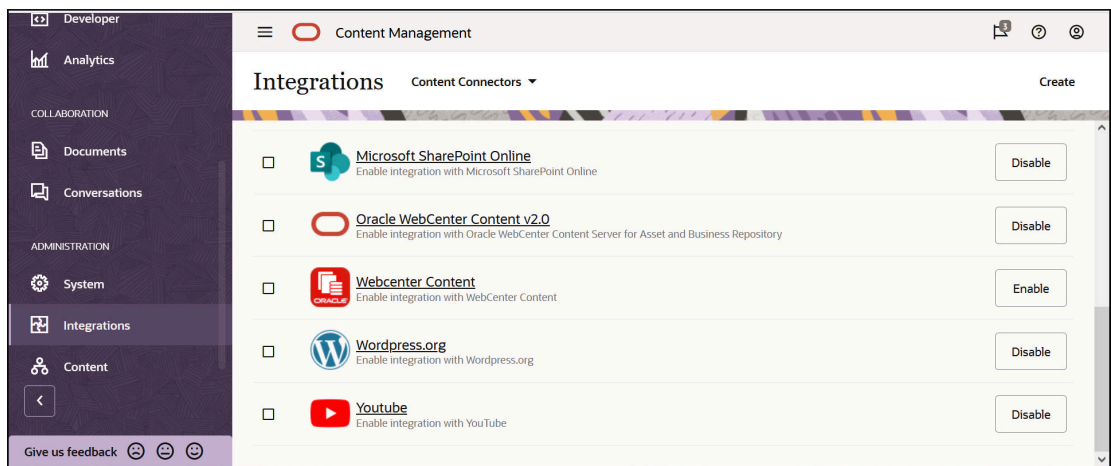
4

Use Content Connectors

You can use content connectors to connect to third-party repositories, such as Google Drive or Dropbox, and import content from the repository into Oracle Content Management.



The screen capture below shows you how to access the content connectors from the Integrations page in the Oracle Content Management web UI. From the left navigation panel, select **Integrations**, then from the top dropdown menu, select **Content Connectors**.



Note:

Content connectors aren't supported in private instances.

Read more related articles here:

- [Enable a Content Connector](#)
- [Disable a Content Connector](#)
- [Configure a Google Drive Content Connector](#)
- [Configure a Microsoft OneDrive Content Connector](#)
- [Configure a Dropbox Content Connector](#)
- [Configure a WordPress.org Content Connector](#)
- [Configure a YouTube Content Connector](#)
- [Configure a Microsoft SharePoint Online Content Connector](#)
- [Configure a Contentful Content Connector](#)
- [Configure a Drupal Content Connector](#)
- [Configure Oracle WebCenter Content Server and Oracle Content Management for the WCC Connector v2.0](#)
- [Use Custom Digital Asset Types in Content Connectors](#)
- [Create and Configure a Custom Content Connector](#)
- [Provide Configuration Parameter Values for a Content Connector](#)
- [Delete a Content Connector](#)

Enable a Content Connector

Oracle Content Management provides preconfigured content connectors for Google Drive, Microsoft OneDrive, Dropbox, Microsoft SharePoint, WebCenter Content, and YouTube. A service administrator can enable any or all of these through the Administration Integrations web interface.

To enable a preconfigured content connector:

1. Sign in to the Oracle Content Management web interface as an administrator or developer.
2. Click **Integrations** in the Administration area of the navigation menu.
3. In the Integrations menu, choose **Content Connectors**.
4. Click **Enable** next to the content connector you want to enable.
5. Change the information and settings as necessary:
 - **Name:** You can change the default content connector name to a user-friendly name.
 - **Description:** You can change or add the description.
 - **Connector service URL:** This read-only field becomes editable if the URL can't connect to the service. Once the URL can connect, it displays as read only again.
 - **Redirect URL:** Make note of the redirect URL in this read-only field. You'll need to specify the URL later for the content connector configuration.
 - **User name:** Enter your administrator user name.

- **User password:** Enter your administrator password.
 - **Connector tags:** You can assign tags that will be applied to assets pulled from the content connector (for example, the content connector name). This lets you search for all items from that content connector in an asset repository.
 - **Enabled for end users:** To enable the content connector, select this check box, or deselect it to disable the content connector.
6. Enter custom field values on the **Additional Fields** tab. How to get the values for the custom fields varies, depending on the content connector:
 - [Configure a Google Drive Content Connector](#)
 - [Configure a Dropbox Content Connector](#)
 - [Configure a Microsoft OneDrive Content Connector](#)
 - [Configure a YouTube Content Connector](#)
 - [Configure a Microsoft SharePoint Online Content Connector](#)
 - [Configure Oracle WebCenter Content Server and Oracle Content Management for the WCC Connector v2.0](#)
 7. Click **Save**.

After a content connector is configured to connect to a third-party content provider, you can enable it by clicking **Enable** next to the connector on the **Content Connectors** page.

Disable a Content Connector

A service administrator can use the connector framework to disable a content connector in an Oracle Content Management instance at any time.

To disable a content connector:

1. Sign in to the Oracle Content Management web interface as an administrator or developer.
2. Click **Integrations** in the Administration area of the navigation menu.
3. In the Integrations menu, choose **Content Connectors**.
4. Click **Disable** next to the content connector you want to disable.

After a content connector is disabled, it's no longer available on the **Add From** drop-down menu on the Assets page, until it is enabled again. You can now click the **Enable** button to enable the content connector again.

Configure a Google Drive Content Connector

After you configure and enable a Google Drive content connector, you can connect to Google Drive and pull required content from it into Oracle Content Management.

To configure a Google Drive content connector:

1. Get the Oracle Content Management host name and authorization URL details.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.

- d. Click the check box next to the Google Drive content connector.
- e. Click **Create**.
- f. On the **General** tab, note the Redirect URL, which will be in this format:

```
https://<hostname>.<domainname>.com:<port>/documents/web/  
AR_COMPLETE_AUTHORIZATION
```

The **Redirect URL** field is truncated. Double-click the field to copy the value to your clipboard.

- 2. Create a Google Drive app:
 - a. Go to <https://console.developers.google.com/project> in a browser window.
 - b. Sign in with a Google user name and password.
If you've already created projects in the Google Developers Console, you will see the list of created projects. If not, the **Manage resources** screen will appear.
 - c. Click **Create Project**.
You'll be redirected to a page where you can enter **Project Name** and **Project ID** values to help you recognize your project in the console.
 - d. Click **Create**.
Your project will be created, and you'll be redirected to your projects list in the console.
 - e. Go to the dashboard through menu navigation, **APIs & Services > Dashboard**, from the left corner icon.
 - f. Click the name of the project you recently created to go to your project dashboard.
 - g. Select a library. Go to Library page through menu navigation, **APIs & Services > Library**, from the left corner icon.
 - h. Search for the "Google Drive API" library, then select and enable it.
 - i. Search for the "Google Picker API" library, then select and enable it.
 - j. From the left corner icon, navigate to **APIs & Services > Credentials > OAuth Consent Screen** tab. Check your email address and enter your product name, **GDrivePickerApp**, and save it.
 - k. Publish the application.
 - l. Click the **Credentials** tab to open a popup in which you can select an OAuth client ID option under the **Create credentials** select box.
 - m. Select your application type, **Web application**.
 - n. Name the client and add the Authorized redirect URIs.
 - i. In **Authorized redirect URI**, paste the redirect URL you copied from Oracle Content Management.
The redirect URI must be the same as your application installation URL.
 - ii. In **Authorized Javascript Origins**, paste the redirect URL, but remove everything after the port.
If you get the message "Invalid Origin", click the authorized domains list and add your domain.

- iii. Press ENTER, and then click **Save**.

Once you have added to the authorized domains, you should be able to create an OAuth ID. Note the values of Client ID and Client Secret.

3. Create credentials to get the API key (Developer Key). Click **Create Credentials** and then **API key** in the drop-down menu. Note the value of API Key. This is the Developer Key you'll enter in the Oracle Content Management when you configure the Google Drive connector.
4. Get the App ID.
 - a. From the left corner icon, navigate to **IAM & admin > Settings**.
 - b. From that page note the value of Project Number. This is the App ID you'll enter in the Oracle Content Management when you configure the Google Drive connector.
5. Configure the connector in Oracle Content Management.
 - a. Go back to configuring the Google Drive content connector in Oracle Content Management.
 - b. On the **General** tab:
 - In the **Connector tags** field, you can assign tags that will be applied to assets pulled from the content connector (for example, the content connector name). This lets you search for all items from that content connector in an asset repository.
 - Make sure **Enabled for end users** is selected. You can review the **Terms of use** and **Privacy Policy** below the **Enable for end users** button.
 - c. Click the **Additional Fields** tab, and enter the following information:
 - **Client ID**
 - **Client Secret**
 - **Developer Key** (API Key)
 - **App Id** (Project Number)
 - d. When you're done, click **Save**.
6. Associate the connector with one or more asset repositories.
 - a. Click **Assets** in the Administration area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.
 - d. Under **Connectors**, select one or more connectors to associate with the repository. This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance.

If any of the connectors you select have content types associated with them, the types appear under **Content Types**.

After a Google Drive content connector is enabled, configured, and associated with a repository, it's available in the asset repository for Oracle Content Management users to import content, through the **Add** drop-down menu, **Import from Google Drive** option on the **Assets** page.

Configure a Microsoft OneDrive Content Connector

After you configure and enable a Microsoft OneDrive content connector, you can connect to OneDrive and pull required content from it into Oracle Content Management.

A Microsoft OneDrive content connector enables you to import assets from the OneDrive repository into Oracle Content Management. As an Administrator or Developer, you can configure a Microsoft OneDrive content connector from the **Integrations** menu in an Oracle Content Management instance.

To configure a OneDrive content connector:

1. Get the Oracle Content Management host name and authorization URL details.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the OneDrive content connector.
 - e. Click **Create**.
 - f. On the **General** tab, note the Redirect URL, which will be in this format:

```
https://<hostname>.<domainname>.com:<port>/documents/web/  
AR_COMPLETE_AUTHORIZATION
```

The **Redirect URL** field is truncated. Double-click the field to copy the value to your clipboard. You'll need to provide this value later in the **OneDrive URL** field on the **Additional Fields** tab.

2. Create a OneDrive app.
 - a. Sign in to Microsoft Azure at <https://portal.azure.com/>:
 - i. Pick an account that has your Microsoft user name.
 - ii. Enter your Microsoft password.
 - iii. Choose **Office365**.
 - iv. Click **Sign in**.
 - v. Click **Yes** to stay signed in and reduce multiple sign-ins.
 - b. On the **Azure services** page, click **App registrations**.
 - c. On the **App registrations** page, click **New registration**.
 - d. On the **Register an application** page, enter a name and choose the account types that can use the application. At the bottom, paste a redirect URL and then click **Register**.
 - e. On your application's page, click **Certificates & secrets** in the left menu, and then click the **New client secret** button on the page.
 - f. Under **Add a client secret**, enter a description for your client secret, select **Never** under **Expires**, and then click **Add**.

- g. Click the **Copy to clipboard** icon on the bottom right to record the secret. You'll need to enter it later in the **Client Secret** field on the **Additional Fields** page for the content connector. Also click **Copy to clipboard** to record other details.
 - h. On the **API permissions** page, click **Add a permission** to add a OneDrive permission, such as **User.Read**, a **Delegated** permission for OneDrive to sign in as the user and access an API. Another permission you might want to add is **off-line access**, which enables the refresh token.
In addition to **User** permissions, you can grant **AllSites**, **Myfiles**, **Sites**, and **TermStore** permissions.
 - i. Click **Add an app**.
 - j. Enter an application name, and click **Create**. The sampleApp Registration screen is displayed.
 - k. Under **Properties**, type the name of your application.
 - l. Note the **Application id** value.
3. Add the OneDrive connector to a repository..
 - a. In Oracle Content Management, select **Integrations** on the left navigation menu.
 - b. On the **Integrations** page, choose **Content Connectors** from the drop-down menu.
 - c. In the list of content connectors, click the name of the **OneDrive** content connector.
 4. Configure the Microsoft OneDrive content connector in Oracle Content Management.
 - a. On the **General** tab:
 - The **Name**, **Description**, **Connector service URL**, and **Redirect URL** fields are prepopulated.
 - In the **Connector tags** field, you can assign tags that will be applied to assets pulled from the content connector (for example, the content connector name). This lets you search for all items from that content connector in an asset repository.
 - Make sure **Enabled for end users** is selected.
You can review the **Terms of use** and **Privacy Policy** below the **Enable for end users** button.
 - b. Click the **Additional Fields** tab, and enter the following information:
 - **Client ID** (the value of **Application id** that you noted down earlier)
 - **Client Secret** (the value of the generated password)
 - **OneDrive URL**
 - **Tenant ID**
 - **Access Scope**
 - c. When you're done, click **Save**.
 5. Add the connector to a repository.
 - a. Click **Content** in the **Administration** area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.

- d. Under **Content Connectors**, select one or more connectors to associate with the repository.
This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance.

If any of the connectors you select have content types associated with them, the types appear under **Content Types**.

After a Microsoft OneDrive content connector is configured, enabled, and associated with a repository, it's available in the asset repository for Oracle Content Management users to download content, through the **Add** drop-down menu **Import from Microsoft OneDrive** option on the **Assets** page.



Note:

An authorized application for a OneDrive content connector must provide a link to <https://account.live.com/consent/Manage>, or to another location specified for a OneDrive content connector, with a clear indication that end users can go to that Microsoft site to revoke permissions at any time. If end users must take additional steps to disable the authorized application's access to end-user information, then the application must clearly indicate to end users the additional steps required to disable access. These requirements do not apply where Microsoft provides the end-user web interface.

Configure a Dropbox Content Connector

After you configure and enable a Dropbox content connector, you can connect to Dropbox and pull required content from it into Oracle Content Management.

To configure a Dropbox content connector:

1. Get the Oracle Content Management host name and authorization URL details.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the Dropbox content connector.
 - e. Click **Edit**.
 - f. On the **General** tab, note the Redirect URL, which will be in this format:

```
https://<hostname>.<domainname>.com:<port>/documents/web/  
AR_COMPLETE_AUTHORIZATION
```

The **Redirect URL** field is truncated. Double-click the field to copy the value to your clipboard.

2. Create a Dropbox app:
 - a. Go to <https://www.dropbox.com/developers/apps> in a Browser window
 - b. Sign in with a Dropbox user name and password.

- c. Go to **My apps**.
 - d. Click **Create App**, and on next screen, in Step 1, choose a Dropbox API.
 - e. Choose an access type and name for your application, and click **Create App**. For production deployment of the Dropbox app, refer to Dropbox documentation for the approval required: <https://www.dropbox.com/developers/reference/developer-guide#production-approval>.
 - f. After the app is created, you can add or configure additional values on the next screen.
 - g. Note the values of App key and App secret.
3. In the **OAuth 2 - Redirect URIs** section, paste the Redirect URL you copied from Oracle Content Management.
 4. On the **Dropbox App** page, navigate to the lower section, **Chooser/Saver domains**, and add the host name used to access Oracle Content Management to this section.
 5. Configure the connector in Oracle Content Management.
 - a. Go back to configuring the Dropbox content connector in Oracle Content Management.
 - b. On the **General** tab:
 - In the **Connector tags** field, you can assign tags that will be applied to assets pulled from the content connector (for example, the content connector name). This lets you search for all items from that content connector in an asset repository.
 - Make sure **Enabled for end users** is selected. You can review the **Terms of use** and **Privacy Policy** below the **Enable for end users** button.
 - c. Click the **Additional Fields** tab, and enter the following information:
 - **App key**
 - **App secret**
 - d. When you're done, click **Save**.
 6. Associate the connector with one or more asset repositories.
 - a. Click **Assets** in the Administration area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.
 - d. Under **Connectors**, select one or more connectors to associate with the repository. This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance.

If any of the connectors you select have content types associated with them, the types appear under **Content Types**.

After a Dropbox content connector is configured, enabled, and associated with a repository, it's available in the asset repository for Oracle Content Management users to download content, through the **Add** drop-down menu, **Import from Dropbox** option on the **Assets** page.

Enable Jax-WS

JAX-WS should be enabled on Oracle Content Server for the connector to work. This is the basic prerequisite.

JAX-WS should be enabled on Oracle Content Server for the connector to work. This is the basic prerequisite. To verify this, access the content server instance using the following URL. This will render the WSDL in the browser.

```
https://<host_name>:<port>/idcnativevs/IdcWebLoginPort?WSDL
```

Configure a WordPress.org Content Connector

After you configure and enable a WordPress.org content connector, you can connect to WordPress.org and pull required content from it into Oracle Content Management.

To create and configure a custom content connector:

1. Locate the base URL for your WordPress.org instance. For example, `https://wordpress.mycompany.com/`.
2. Create the connector in Oracle Content Management.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the WordPress.org content connector and then click **Create**.
 - e. On the Connector Settings page, provide information for registration of your WordPress.org content connector. If you want to accept a certificate signed by the third-party provider, select **Accept self-signed certificate**.
 - f. Click **Next**. Once the details are verified, select the Custom Fields tab.
 - g. Click the Custom Fields tab, and fill the **WordPress URL** field.
3. When you are done, click **Save**.
4. On the Content Connectors page, click the **Enable** button next to the WordPress.org content connector you created.
5. Associate the connector with one or more asset repositories:
 - a. Click **Content** in the Administration area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.
 - d. Under **Content Connectors**, select one or more connectors to associate with the repository.

This menu lists all the content connectors that have been configured and enabled in your Oracle Content and Experience instance. If any of the

connectors have content types associated with them, the types appear under **Content Types**.

After a WordPress.org content connector is enabled, configured, and associated with a repository, it's available in the asset repository for Oracle Content and Experience users to import content. To import content, click the **Add** drop-down menu and select **Import from WordPress.org** option on the Assets page.

Configure a YouTube Content Connector

After you configure and enable a YouTube content connector, you can connect to YouTube and pull required content from it into Oracle Content Management.

To configure a YouTube content connector:

1. Get the Oracle Content Management host name and authorization URL details.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the **Youtube** content connector.
 - e. Click **Create**.
 - f. You can override the default content connector settings, as described in [Enable a Content Connector](#).
2. Enter any name and give the Connector Service URL.
3. Click **Verify Settings**.

This lists the custom fields of the YouTube content connector.

In the **Connector Tags** field, you can assign tags for assets that the content connector pulls into Oracle Content Management.
4. Sign in to <https://console.developers.google.com/> and create a new project. Give your project a name, and click **CREATE**.
5. Go to the project and the **Library** page from the left corner icon in the navigation menu, and click **ENABLE APIS AND SERVICES**.
6. Search for the **Youtube Data API v3** library, select it, and enable it.
7. Click **Create Credentials** and then **API key** in the drop-down menu. Note the value of **API Key**. This is the **Developer Key** you'll enter in Oracle Content Management when you configure the YouTube connector.
8. Specify a value for **Application Name**, which can be anything.
9. Click **Save**.
10. Make sure **Enabled for end users** is selected.

You can review the **Terms of use** and **Privacy Policy** below the **Enable for end users** button.
11. Map source metadata to content types for the connector.

When you configure a content connector, you can do field mapping from source metadata to content type fields. A repository administrator or system administrator can allow creation of content types for a connector. You can create content types on the **Mappings** tab of the **Connector Settings** page. To map source metadata to a content type:

- a. Sign in to Oracle Content Management as a system administrator and click **Integrations** under **Administration** in the left navigation menu.
 - b. On the **Assets** page, click **Create**.
 - c. On the **Create Repository** page, define your repository.
 - i. Specify the connector name, publishing channels, languages, and other options.
 - ii. Under the **Connectors** option, select **Youtube** from the drop-down menu to associate the YouTube content connector with the repository. On the **Create/Edit Repository** page, if you add the connector 'Youtube', the content type 'YoutubeVideos' will automatically be assigned. If the YouTube connector has content types associated with it, the types appear under the **Content Types** option. If you want the content connector to have additional fields, you can add them on the **Mappings** tab of the **Connector Settings** page.

If the **Allow Content Type Creation** check box on the **General** tab is selected, the mapping of source metadata to Oracle Content Management content types will be automatically populated. You can change the automatic mapping on the **Connector Settings** page.
 - d. If the **Allow Content Type Creation** check box on the **General** tab is not selected, you need to map the source metadata to Oracle Content Management content types on the **Connector Settings** page:
 - i. On the **Mappings** tab, choose a source type from the **Source Content Type** drop-down menu and an Oracle Content Management type from the **OCE Content Type** drop-down menu, such as **YoutubeVideos**, to map the **Source Data Field** to the **Target Data Field**.
 - ii. Repeat for each source content type.
 - iii. Click **Save**.
 - iv. [Enable a Content Connector](#).

This type mapping must be done for all source types before the content connector is enabled. If you want to change the mapping, you need to disable the connector, make your changes, and then enable the connector again. Whenever an administrator saves the connector after modifying a content type, the connector framework will attempt to seed the content type again. If you remove any associated content type that is required for the selected connector, the save will fail.
12. Associate the connector with one or more asset repositories.
- a. Click **Assets** in the **Administration** area of the navigation menu on the left.
 - b. Open an existing repository, or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options. You need to configure the languages that are expected from the connector. For example, some of the languages that YouTube supports are **en**, **en-CB**, and **en-AU**. If these are not configured, content item creation may fail.
 - d. Under **Connectors**, select **Youtube** to associate the YouTube content connector with the repository.

This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance.

If any of the connectors you select have content types associated with them, the types appear under **Content Types**. On the **Create/Edit Repository** page, if you add the connector 'Youtube', the content type 'YoutubeVideos' will be automatically assigned.

Configure a Microsoft SharePoint Online Content Connector

Microsoft SharePoint Online is a web-based document management and storage system. After you configure and enable a Microsoft SharePoint Online content connector, you can connect to it and pull content from Microsoft SharePoint Online into Oracle Content Management.

A Microsoft SharePoint Online content connector enables you to import assets from the Microsoft SharePoint Online repository into Oracle Content Management. As an Administrator or Developer, you can configure a Microsoft SharePoint Online Content Connector from the Integrations menu in an Oracle Content Management instance.

To configure a Microsoft SharePoint Online content connector:

1. Get the Oracle Content Management host name and authorization URL details.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the Microsoft SharePoint Online content connector.
 - e. Click **Edit**, or you can click the link to get to the **Edit** page.
 - f. On the **General** tab, note the Redirect URL, which will be in this format:

```
https://<hostname>.<domainname>.com:<port>/documents/web/
AR_COMPLETE_AUTHORIZATION
```

The **Redirect URL** field is truncated. Double-click the field to copy the value to your clipboard. You'll need to provide this value later on the Azure app registration page.

2. Create a SharePoint Online app:
 - a. Sign in to Microsoft Azure at <https://portal.azure.com/>:
 - i. Pick an account that has your Microsoft user name.
 - ii. Enter your Microsoft password.
 - iii. Choose **Office365**.
 - iv. Click **Sign in**.
 - v. Click **Yes** to stay signed in and reduce multiple sign-ins.
 - b. On the **Azure services** page, click **App registrations**.
 - c. On the **App registrations** page, click **New registration**.
 - d. On the **Register an application** page, enter a name and choose the account types that can use the application. At the bottom, paste a redirect URL and then click **Register**.
 - e. On your application's page, click **Certificates & secrets** in the left menu, and then click the **New client secret** button on the page.

- f. Under **Add a client secret**, enter a description for your client secret, select **Never** under **Expires**, and then click **Add**.
 - g. Click the **Copy to clipboard** icon on the bottom right to record the secret. You'll need to enter it later in the **Client Secret** field on the **Additional Fields** page for the content connector. Also click **Copy to clipboard** to record other details.
 - h. On the **API permissions** page, click **Add a permission** to add a SharePoint Online permission, such as **User.Read**, a **Delegated** permission for SharePoint Online to sign in as the user and access an API. Another permission you might want to add is **off-line access**, which enables the refresh token.
In addition to **User** permissions, you can grant **AllSites**, **Myfiles**, **Sites**, and **TermStore** permissions.
3. Add the SharePoint Online connector to a repository.
 - a. In Oracle Content Management, select **Integrations** on the left navigation menu.
 - b. On the **Integrations** page, choose **Content Connectors** from the drop-down menu.
 - c. In the list of content connectors, click the name of the **SharePoint Online** content connector.
 4. Configure the Microsoft SharePoint Online content connector in Oracle Content Management.
 - a. On the **General** tab:
 - The **Name**, **Description**, **Connector service URL**, and **Redirect URL** fields are prepopulated.
 - In the **Connector tags** field, you can assign tags that will be applied to assets pulled from the content connector (for example, the content connector name). This lets you search for all items from that content connector in an asset repository.
 - Make sure **Enabled for end users** is selected. You can review the **Terms of use** and **Privacy Policy** below the **Enable for end users** button.
 - b. Click the **Additional Fields** tab.

The screenshot shows the Oracle Content Management interface for configuring a Microsoft SharePoint connector. The left navigation pane includes Home, Assets, Sites, Experiences, Recommendations, Developer, Analytics, and a COLLABORATION section with Documents, Conversations, and Sauce Video. The main content area is titled 'Microsoft SharePoint' and has 'Cancel' and 'Save' buttons. The 'Additional Fields' tab is selected, showing five input fields:

- * Client ID
- * Client Secret
- * SharePoint URL
- * Tenant ID
- * Access Scope

Enter the values for the following fields, some of which you recorded earlier:

- **Client ID:** the value of the **Application ID** from the Azure app registration.
 - **Client Secret:** the value of the client secret generated in the Azure app registration.
 - **SharePoint URL:** `https://<company name>.sharepoint.com`, this is the SharePoint URL, URL of your tenant, and URL of site collection.
 - **Tenant ID:** the **Directory (tenant) ID** from the Azure app registration.
 - **Access Scope:** `offline_access https://<company name>.sharepoint.com/AllSites.Read`
- c. When you're done, click **Save**.
5. Add the connector to a repository.
- a. Click **Content** in the **Administration** area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.
 - d. Under **Content Connectors**, select one or more connectors to associate with the repository.
This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance.
If any of the connectors you select have content types associated with them, the types appear under **Content Types**.

After a Microsoft SharePoint Online content connector is configured, enabled, and associated with a repository, it's available in the asset repository for Oracle Content Management users to download content, through the **Add** drop-down menu **Import from Microsoft SharePoint** option on the **Assets** page.

**Note:**

An authorized application for a Microsoft SharePoint Online content connector must provide a link to `https://account.live.com/consent/Manage`, or to another location specified for a SharePoint Online content connector, with a clear indication that end users can go to that Microsoft site to revoke permissions at any time. If end users must take additional steps to disable the authorized application's access to end-user information, then the application must clearly indicate to end users the additional steps required to disable access. These requirements do not apply where Microsoft provides the end-user web interface.

Configure a Contentful Content Connector

After you configure and enable a Contentful content connector, you can connect to Contentful and pull required content from it into Oracle Content Management.

To configure a Contentful content connector:

1. Get the Contentful API key:
 - a. Locate the base URL for your Contentful instance. For example, `https://contentful.company.example.com/`.

- b. Click **Settings** in the top menu. Click **API keys**. Click **Add API key**.
 - c. Add a name to the **Name** field.
 - d. Note the Space ID, Content Delivery API Access Token, Content Preview API Access Token, and Environment. You will need them later.
 - e. Click **Save**.
2. Configure the Contentful connector in Oracle Content Management.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the Contentful content connector, and then click **Edit**.
 - e. On the **Connector Settings** page, on the **General** tab, in the **Connector tags** field, you can assign tags that will be applied to assets pulled from the content connector (for example, the content connector name).

This lets you search for all items from that content connector in an asset repository.

Make sure **Enabled for end users** is selected. You can review the Terms of use and Privacy Policy below the **Enabled for end users** button.
 - f. Click the **Additional Fields** tab, and enter the following information:

If you want to import Draft content, select the **Draft Content** check box and set **Authentication Key** to the Content Preview API Access Token. Otherwise, enter the Content Delivery API Access Token.

Enter your Space ID and your Environment.

After a Contentful content connector is enabled, configured, and associated with a repository, it's available in the asset repository for Oracle Content Management users to import content. To import content, click the **Add** drop-down menu and select the **Import from Contentful** option on the **Assets** page.

Configure a Drupal Content Connector

After you configure and enable a Drupal content connector, you can connect to Drupal and pull required content from it into Oracle Content Management.

To configure a Drupal content connector:

1. Sign in to the Drupal site.
 - a. Click **Extend** on the menu.
 - b. On the Extend page, search for JSON and then select **JSON:API** and **Serialization** checkboxes.
 - c. Select **HTTP Basic Authentication**.
 - i. You must have the HTTP Basic Authentication module enabled.
 - ii. Click **People**.
 - iii. Click **Permissions**.

- iv. Dselect checkboxes under Anonymous to protect resources with Basic Authenticaiton.
 - d. Click **Install**.
2. Locate the base URL for your Drupal instance. For example, *https://drupal.mycompany.com/*.
3. Get the Oracle Content Management host name and authorization URL details.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click the check box next to the Drupal content connector and then click **Create**.
 - e. On the Connector Settings page, provide information for registration of your Drupal content connector.
 - f. Click **Next**. Once the details are verified, select the Custom Fields tab.
 - g. On the Custom Fields page, enter the base URL of Drupal instance and click **Save**.
4. On the Content Connectors page, click the **Enable** button next to the Drupal content connector you created.
5. Associate the connector with one or more asset repositories:
 - a. Click **Content** in the Administration area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.
 - d. Under **Content Connectors**, select one or more connectors to associate with the repository.
This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance. If any of the connectors have content types associated with them, the types appear under **Content Types**.

After a Drupal content connector is enabled, configured, and associated with a repository, it's available in the asset repository for Oracle Content Management users to import content. To import content, click the **Add** drop-down menu and then select **Import from Drupal** option on the Assets page.

Configure Oracle WebCenter Content Server and Oracle Content Management for the WCC Connector v2.0

An Administrator can register and configure Oracle WebCenter Content (WCC) and Oracle Content Management for the WCC connector v2.0 so site users can pull content from the WebCenter Content server into their sites.

The following topics describe the WebCenter Content connector v2.0 and how to use it:

- [Verify Network Accessibility for a WCC Connector v2.0](#)
- [Check WebCenter Content Server Readiness](#)
- [Check Oracle Content Management Readiness](#)
- [Configure the Oracle WebCenter Content Connector v2.0](#)

- [Specify WebCenter Content Connector v2.0 Mappings](#)
- [Enable Oracle WebCenter Content Connector v2.0 for an Asset Repository](#)
- [Use Oracle WebCenter Content Connector v2.0](#)
- [Map to a Custom Asset Type](#)
- [Revoke Authorized Users](#)

Verify Network Accessibility for a WCC Connector v2.0

The communication between Oracle Content Management and WebCenter Content Server takes place over JAX-WS protocol, so the Oracle Content Management instance and WebCenter Content Server need to be on the same network for them to work together.

- For the Oracle Content Management public cloud, the WebCenter Content Server should be accessible from the Public networks.
- For the Oracle Content Management private cloud, the WebCenter Content Server should be accessible from that private network.

Check WebCenter Content Server Readiness

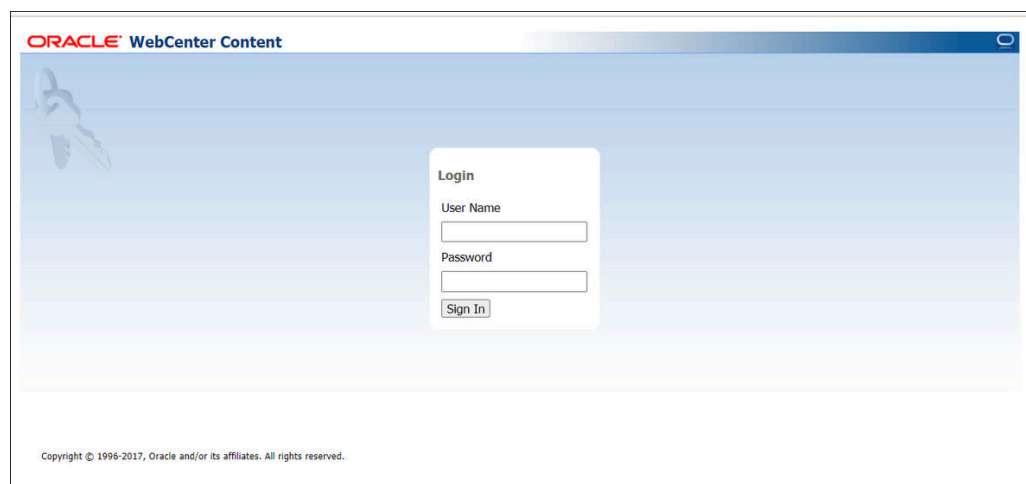
- [Enable SSL](#)
- [Enable Jax-WS](#)
- [Set Up a Security Policy](#)
- [Run the Indexer and File Formats Wizard](#)

Enable SSL

The Oracle Content Server should be accessible over HTTPS protocol.

Access the content server sign-in page using the following URL, which will render the sign-in page in the browser:

Login Page: `https://<host_name>:<port>/cs/login/login.htm`



Enable Jax-WS

JAX-WS should be enabled on Oracle Content Server for the connector to work. This is the basic prerequisite.

JAX-WS should be enabled on Oracle Content Server for the connector to work. This is the basic prerequisite. To verify this, access the content server instance using the following URL. This will render the WSDL in the browser.

```
https://<host_name>:<port>/idcnativews/IdcWebLoginPort?WSDL
```

Set Up a Security Policy

We will communicate with the WebCenter Content Server using the JAX-WS communication channel. The JAX-WS endpoints must be secured using the OWSM policies.

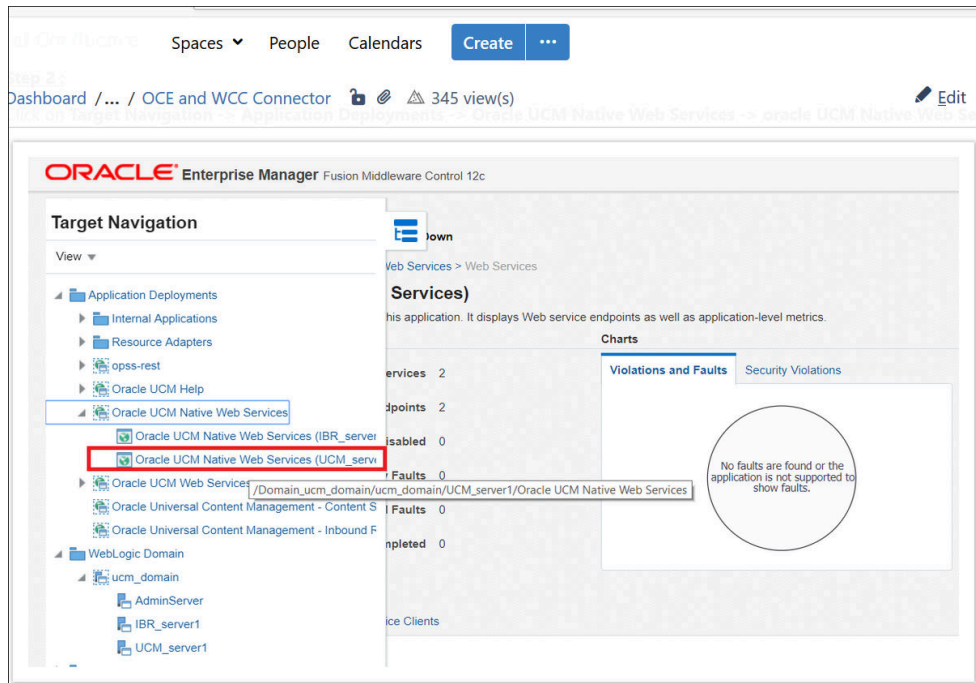
We will use the **oracle/wss_username_token_over_ssl_service_policy** server-side policy. Any client that communicates with the WebCenter Content Server must enter both a user name and a password. If these values are incorrect, then the handshake with the WebCenter Content Server will not be established.

Take the following steps to configure the server-side policy:

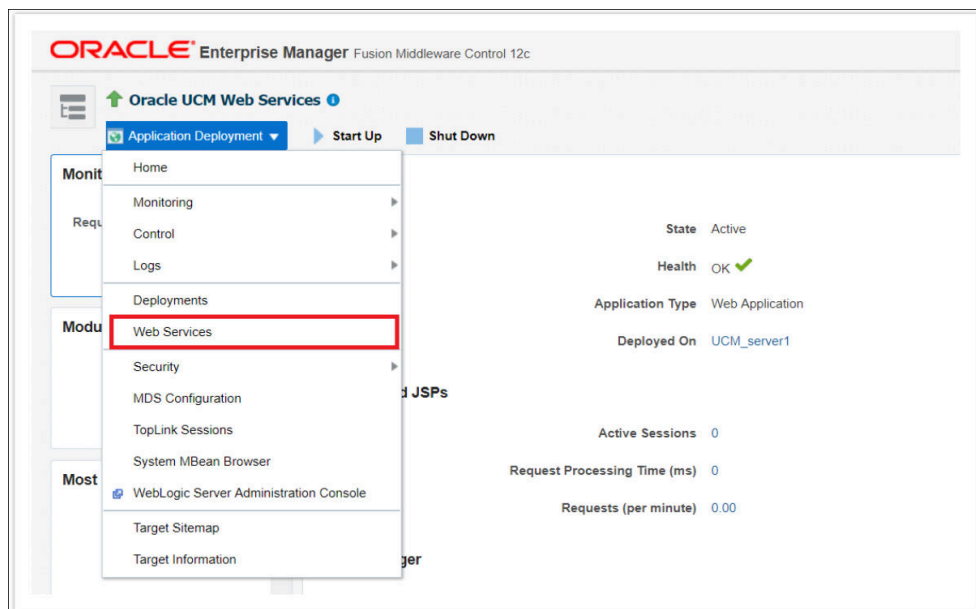
1. Check the **OWSM policy** attached to JAX-WS.
To check the policy attached, access `https://<host_name>:<port>/idcnativews/IdcWebLoginPort?WSDL`. You should see the policy name in the **URI** attribute of the **<wsp:PolicyReference>** field in the WSDL.

```
<binding name="IdcWebLoginSoapHttp" type="tns:IdcWebLogin">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsp:PolicyReference xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    URI="#wss_username_token_over_ssl_service_policy" wsdl:required="false"/>
</binding>
```

2. If the attached policy is **oracle/wss_username_token_over_ssl_service_policy**, then ignore the steps that follow.



3. If the attached policy is a different one, then first detach that policy, and then attach the policy **oracle/wss_username_token_over_ssl_service_policy**. The steps to attach or detach a policy follow. If no policy is attached, then follow the steps.



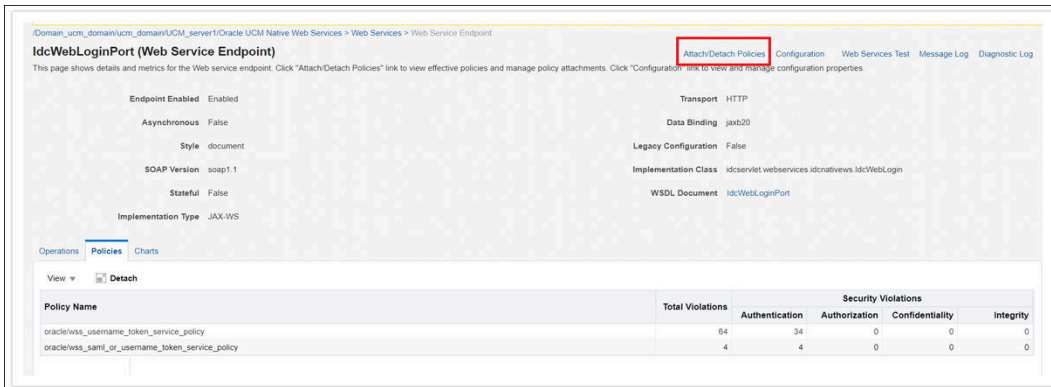
Take the following steps to Attach an OWSM Policy:

1. Go to the WebCenter Content Server Enterprise Manager:

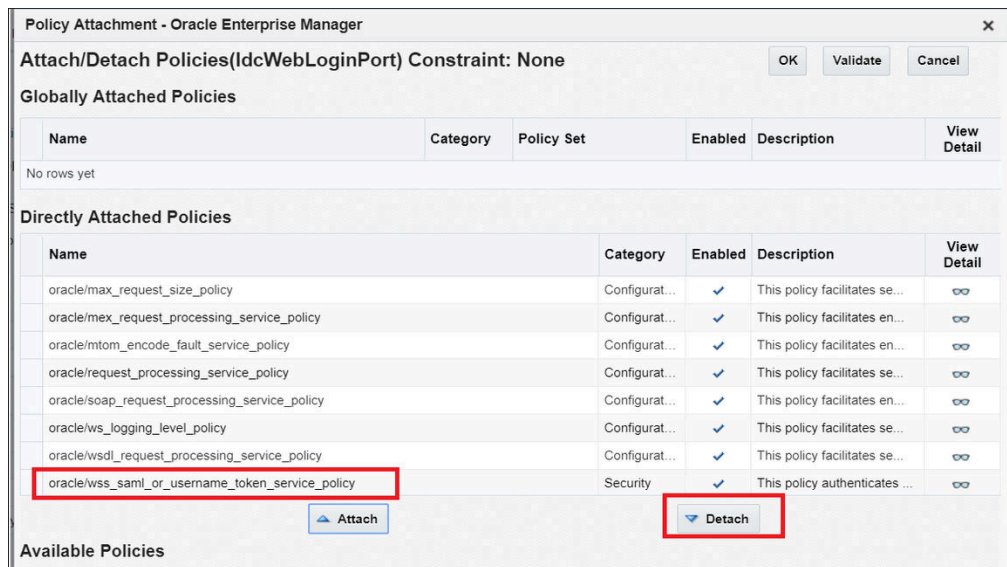
http://<host_name>:<port>/em

Sign in with your administrator credentials (weblogic/<password>)

2. On the **Target Navigation** page, choose **Application Deployments -> Oracle UCM Native Web Services -> oracle UCM Native Web Service(UCM_Server1)**.
3. Choose **Application Deployment -> Web Services**.
4. On the **IdcWebLoginPort (Web Services Endpoint)** page, click **Attach/Detach Policies**.



5. In the directly attached policies section, if there is any policy other than **oracle/wss_username_token_over_ssl_service_policy** attached, first detach it. Then attach the **oracle/wss_username_token_over_ssl_service_policy** policy:
 - a. Click the policy whose category is **Security**, select it, and then click **Detach**. Notice that this policy will go away from the **Directly Attached Policies** list.



- b. Attach the **oracle/wss_username_token_over_ssl_service_policy**. Under **Available Policies**, search for **oracle/wss_username_token_over_ssl_service_policy**. In the result section, select **oracle/wss_username_token_over_ssl_service_policy**, and then click the **Attach** button.

Name	Category	Enabled	Description	View Detail
oracle/max_request_size_policy	Configuration	✓	This policy facilitates se...	∞
oracle/mex_request_processing_service_policy	Configuration	✓	This policy facilitates en...	∞
oracle/mtom_encode_fault_service_policy	Configuration	✓	This policy facilitates en...	∞
oracle/request_processing_service_policy	Configuration	✓	This policy facilitates se...	∞
oracle/soap_request_processing_service_policy	Configuration	✓	This policy facilitates en...	∞
oracle/ws_logging_level_policy	Configuration	✓	This policy facilitates se...	∞
oracle/wsd_request_processing_service_policy	Configuration	✓	This policy facilitates se...	∞

Available Policies

View

Name	Category	Status	Description	View Detail
oracle/wss_username_token_over_ssl_service_policy	Security	✓	This policy uses the crede...	∞

Rows Selected 1 Showing 1 out of 115 Rows

6. Verify that **oracle/wss_username_token_over_ssl_service_policy** is under **Directly Attached Policies**, click **Validate**, and then click **OK**. After the popup closes, click the **Return** button. No restart is required.

Name	Category	Policy Set	Enabled	Description	View Detail
No rows yet					

Directly Attached Policies

Name	Category	Enabled	Description	View Detail
oracle/max_request_size_policy	Configuration	✓	This policy facilitates se...	∞
oracle/mex_request_processing_service_policy	Configuration	✓	This policy facilitates en...	∞
oracle/mtom_encode_fault_service_policy	Configuration	✓	This policy facilitates en...	∞
oracle/request_processing_service_policy	Configuration	✓	This policy facilitates se...	∞
oracle/soap_request_processing_service_policy	Configuration	✓	This policy facilitates en...	∞
oracle/ws_logging_level_policy	Configuration	✓	This policy facilitates se...	∞
oracle/wsd_request_processing_service_policy	Configuration	✓	This policy facilitates se...	∞
oracle/wss_username_token_over_ssl_service_policy	Security	✓	This policy uses the crede...	∞

Available Policies

7. Access **https://<host_name>:<port>/idcnativews/IdcWebLoginPort?WSDL**. You should see the policy **oracle/wss_username_token_over_ssl_service_policy** being attached to the **IdcWebLoginPort** in the **wsu:Id** attribute of the **<wsp:Policy>** field. This confirms that the policy is attached correctly.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tns="http://idcnativews.webservices.idcservlet/" xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
```

```

xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="IdcWebLoginService"
targetNamespace="http://idcnativevs.webservices.idcservlet/">
  <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns="http://
schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
wsu:Id="wss_username_token_over_ssl_service_policy">

```

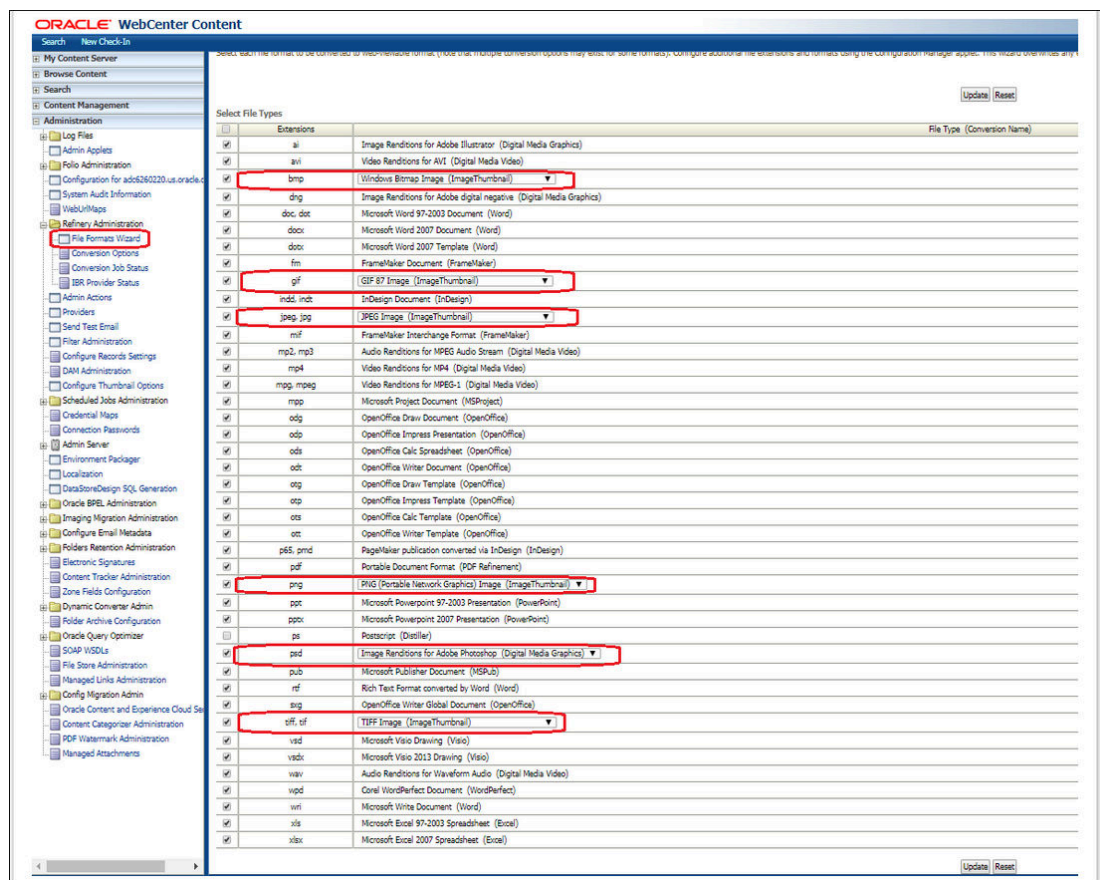
- Also, verify that the **https://<host_name>:<port>/idcws/GenericSoapPort?WSDL** wsdl loads successfully.

Run the Indexer and File Formats Wizard

To complete the configuration of an Oracle WebCenter Content server, you need to run the Indexer for search results and the File Formats Wizard for IBR configuration.

It is mandatory to run the Indexer for a search to return results. Choose **Administration** → **Admin Applets** → **Repository Manager** → **Indexer**.

To show thumbnail for documents in the Oracle WebCenter Content v2.0 file picker window in Oracle Content Management, IBR must be configured for Oracle WebCenter Content Server. Choose **Administration** -> **Refinery Administration** -> **File Formats Wizard**.



Check Oracle Content Management Readiness

The WebCenter Content connector v 2.0 is deployed on an Oracle Content Management (OCM) managed server and can be accessed using the context root: `/cec-connector/wccontent`



Note:

The basic access authentication does not work for federated users. If that's the case, use [token authentication](#) to access the connector.

Basic authentication

To verify the deployment, go to the following link in the browser:

```
http://<host>:<port>/cec-connector/wccontent/rest/api
```

This gives the following response, which indicates the connector is deployed properly:

```
["v1"]
```

Token authentication

1. Log into the OCM service: `https://<host>:<port>/documents` in your web browser.
2. Once you're logged in, open another tab and go to this endpoint to get the token value from the response as shown in your browser.

```
https://<host>:<port>/documents/web?IdcService=GET_OAUTH_TOKEN
```

3. Use the Bearer token value retrieved in step 2 to make a request to the connector endpoint:

```
GET https://<host>:<port>/cec-connector/wccontent/rest/api
```

Make sure to include the authorization field in your header as shown below:

```
Authorization: Bearer <tokenValue>
```

Configure the Oracle WebCenter Content Connector v2.0

Before you start using the Oracle WebCenter Content Connector v2.0 in an Oracle Content Management instance, the connector needs to be registered, configured, and enabled on the Oracle Content Management side.

To configure Oracle WebCenter Content Connector v2.0 for Oracle Content Management:

1. Sign in to Oracle Content Management as an administrator.

- Go to **Administration > Integrations > Content Connectors**, locate the out-of-the-box connector **Oracle WebCenter Content v2.0**, and click **Enable** next to it.

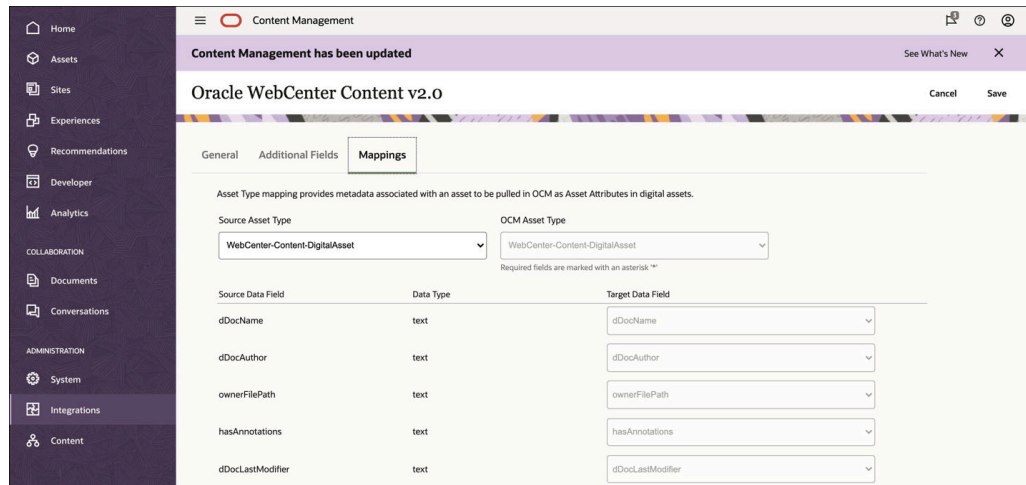
The screenshot shows the Oracle Content Management interface for configuring the Oracle WebCenter Content v2.0 connector. The interface includes a left-hand navigation menu with categories like Home, Assets, Sites, Experiences, Recommendations, Developer, Analytics, COLLABORATION, and ADMINISTRATION. The main content area is titled 'Oracle WebCenter Content v2.0' and has tabs for 'General', 'Additional Fields', and 'Mappings'. The 'General' tab is selected, displaying the following configuration options:

- * Name:** Oracle WebCenter Content v2.0
- Description:** Enable integration with Oracle WebCenter Content Server for Ass
- Connector service URL:** https://Sandbox-occe0002.cec.ocp.oraclecloud.com/cec-connect...
- Allow asset type creation:**
- Connector tags:** Add a tag
- Created:** 7/7/2021 at 8:21 PM
- Last modified:** 11/12/2021 at 11:35 AM
- Enabled for end users:**

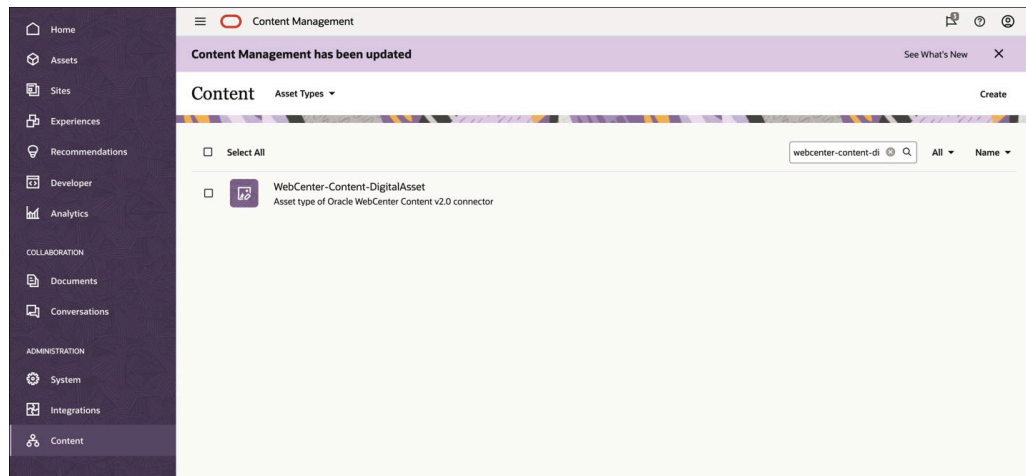
- Enter a description and select the **Allow asset type creation** and **Enable for end-users** checkboxes.
- Enter details for **Additional Fields**:
 - WebCenter Content JAX-WS Connection URI: **http://<host_name>:<port>/idcnativews**
 - WebCenter Content Server JAX-WS Client Policy: **oracle/wss_username_token_over_ssl_client_policy**
 - WebCenter Content Server Web Context Root: **/cs** (This depends on Content Server setup.)
- Click **Save**.

Specify WebCenter Content Connector v2.0 Mappings

Selection of the **Allow asset type creation** option for Oracle WebCenter Content Connector v2.0 creates a **WebCenter-Content-DigitalAsset asset** asset type in Oracle Content Management by the administration user who is configuring the connector.



You can view the asset type on the **Administration > Content > Asset Types** page.

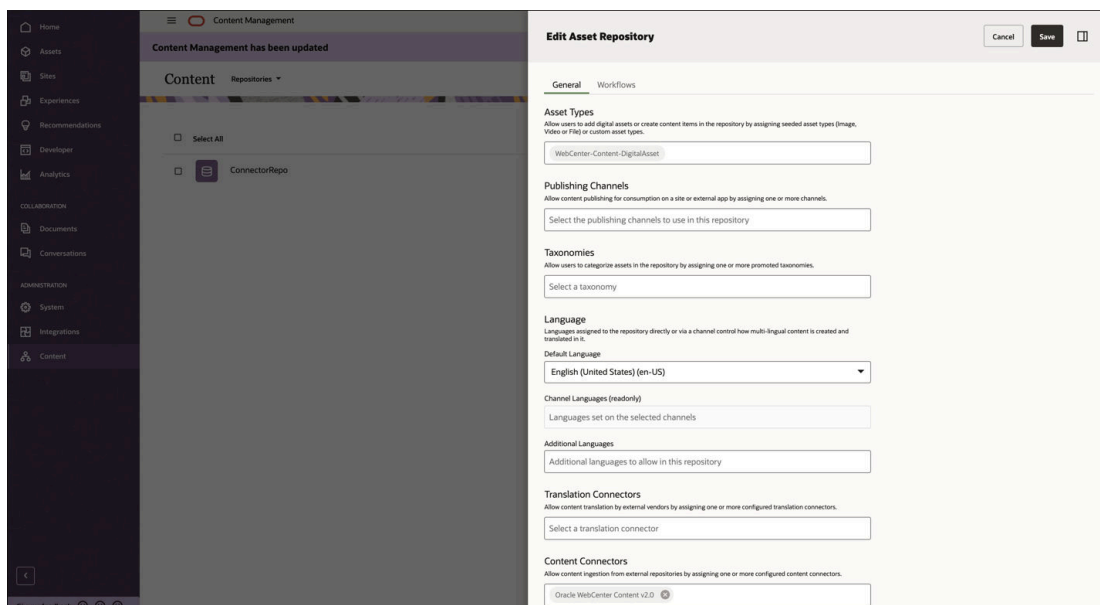


As an administrator for the asset repository, you can specify mappings for the connector type.

Enable Oracle WebCenter Content Connector v2.0 for an Asset Repository

Once the connector is configured and enabled, we need to enable it for the asset repository.

Selecting the connector automatically adds the associated asset type to the repository.



Use Oracle WebCenter Content Connector v2.0

To use Oracle WebCenter Content Connector v2.0, follow these steps:

1. Go to the asset repository for which you have configured the connector, and click **Add > Import from Oracle WebCenter Content v2.0**. This launches a WebCenter Content Sign In dialog.

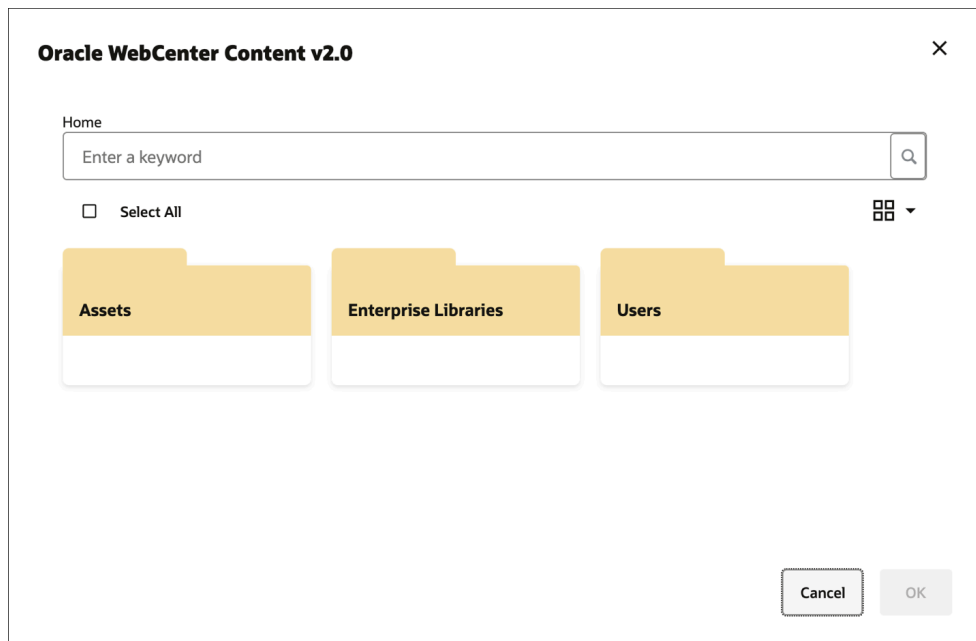
Oracle WebCenter Content v2.0 Sign In ✕

User Name*:

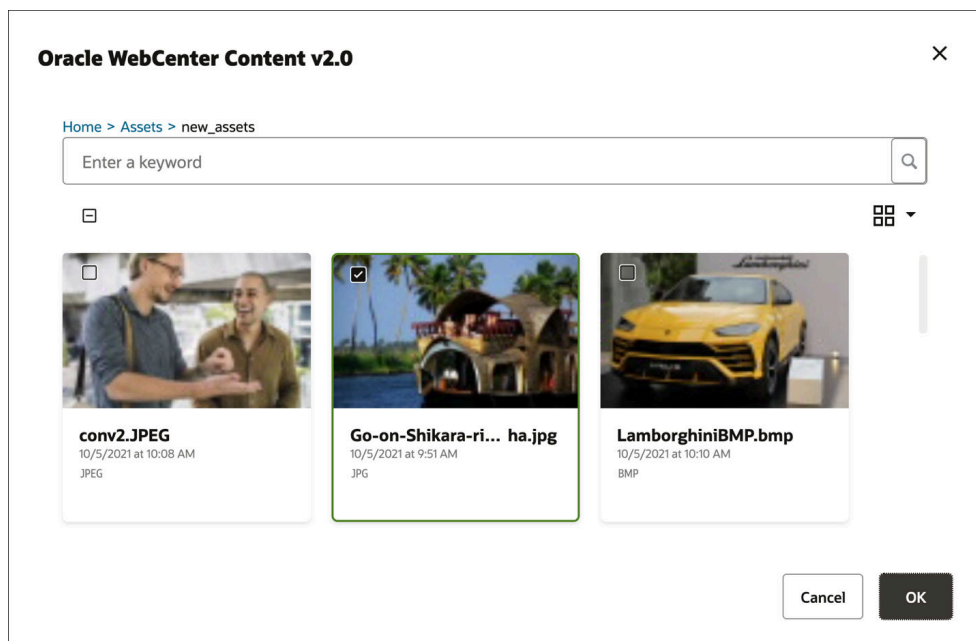
User Password*:

2. Enter your WebCenter Content Server credentials. These credentials will be securely stored on Oracle Content Management and can be reused for subsequent access. To clear the credentials, go to **Add > Manage Sources > Connector > Revoke**.

After your credentials are authenticated, you will see a connector-picker dialog that lists content from the content server.

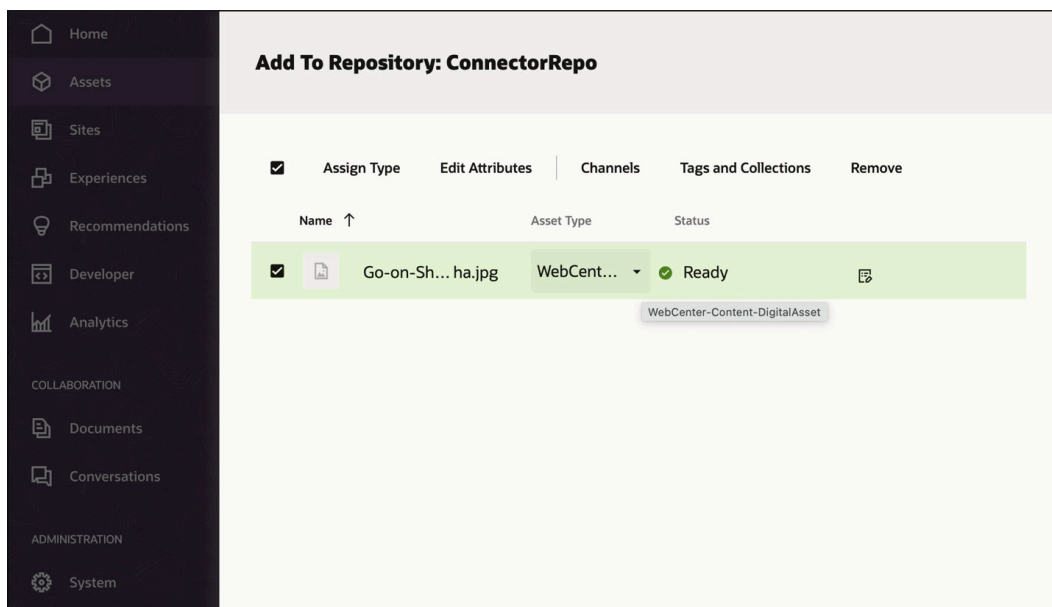


3. Browse through the files and folders. You can also use search for specific items. Additionally, you can choose to view the listing in various layouts, like card, compact, and list.

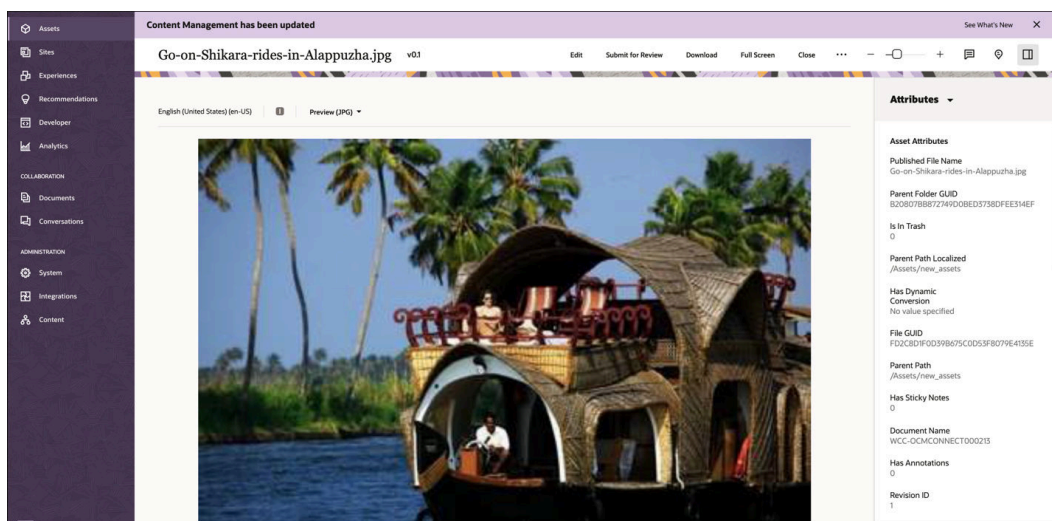


The dialog also supports pagination for a large number of files and folders. Select the file or files you are interested in, and click **OK**.

4. You can add to the asset repository. An intermediate dialog lists the files to be added and lets you choose the collection and channels. You can also specify tags for the content you are adding. Click **Add**. Once **Add** is successful, assets are added in the asset repository.



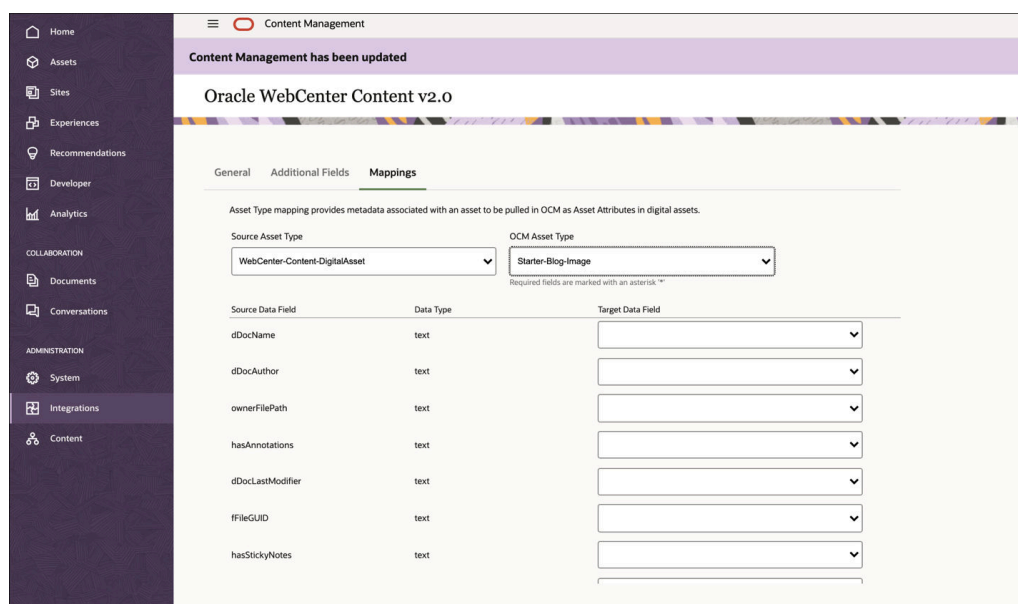
The following image shows the assets and its attributes imported into Oracle Content Management.



Map to a Custom Asset Type

To map a custom asset type, you can reuse an existing asset type or create a new one and do the necessary mapping in the connector configuration screen.

For example, suppose we were to register a new Oracle WebCenter Content v2 connector for another WebCenter Content instance we want to wire to. We deselect **Allow Content type creation** and choose to use other asset type in the Oracle Content Management and perform the necessary mappings of the listed attributes.



Revoke Authorized Users

Once you provide the user credentials, they are stored in Oracle Content Management and passed to the connector during subsequent usage, which means the login challenge will not be given all the time.

If you need to sign in as a different user, then the signed-in authorized user needs to be revoked. To revoke an authorized user:

1. Click **Assets** → **Add** → **Manage Sources**.
2. Click **Revoke** next to a user name.
3. Click **OK** to revoke the user.

Use Custom Digital Asset Types in Content Connectors

You can import into a repository assets that are mapped to digital asset types defined by a content connector.

Import Assets Mapped to Digital Asset Types

Content connectors in Oracle Content Management typically bring assets from a third-party content management system into Oracle Content Management. The assets that are created in Oracle Content Management can be digital assets or content items. You can import assets mapped to asset types defined by a connector into Oracle Content Management.

A digital file as well as the values for custom attributes can come from the external store, such as Google Drive. Now you can import assets into a new repository with seeded types as Image, Video, Video Plus, File, and custom types. You can use a seeded type, a custom digital asset type, or a content item type.

Create and Configure a Custom Content Connector

After you create and configure a custom content connector, you'll be able to connect it to a third-party content repository and import content from the repository into Oracle Content Management.

To create and configure a custom content connector:

1. Create an app for the custom content connector. See [Develop Content Connectors](#).
2. Create the connector in Oracle Content Management.
 - a. Sign in to Oracle Content Management as an Administrator or Developer.
 - b. Click **Integrations** in the **Administration** area of the navigation menu on the left.
 - c. In the **Integrations** menu, choose **Content Connectors**.
 - d. Click **Create**.
 - e. On the **Connector Settings** page, provide information for registration of your custom content connector.
 - f. Click **Next**. Once the details are verified, the **Additional Fields** tab will appear.
 - g. Click the **Additional Fields** tab, and configure the custom fields that were defined for your content connector.

Custom properties are connector-specific. Every connector has its own requirement to connect to a remote store; for example, one connector might need just `ClientID` and `ClientSecret`, while another might require `ClientID`, `ClientSecret`, `AppID`, and so on.
 - h. If the connector's picker type is `CUSTOM`, the following fields are displayed under the **Connector Tags** field:
 - **Custom Picker URL:** Provide the name of the custom picker packaged in your content connector, if any. This setting is not applicable when you use the common web user interface.
 - **Hide OK/Cancel:** This setting indicates whether or not you want Oracle Content Management to embed your picker in their dialog. Leave this setting unchecked if you use the common web user interface.
3. When you're done, click **Save**.
4. If your custom content connector is of type `METADATA` and you want to map source metadata to content type fields, either manually or to edit the automatic mappings, click the **Mappings** tab. You must, [map content types and metadata](#) before enabling the connector.
5. On the **Content Connectors** page, click **Enable** next to the custom content connector you created.
6. Associate the connector with one or more asset repositories.
 - a. Click **Assets** in the Administration area of the navigation menu on the left.
 - b. Open an existing repository or click **Create** to create a new one.
 - c. If you're creating a new repository, specify the repository name, publishing channels, languages, and other options.
 - d. Under **Connectors**, select one or more connectors to associate with the repository.

This menu lists all the content connectors that have been configured and enabled in your Oracle Content Management instance.

If any of the connectors you select have content types associated with them, the types appear under **Content Types**.

Create Content Types for a Connector

A repository administrator or system administrator can allow creation of content types for a content connector.

You can create content types for a connector on the **Mappings** tab of the **Connector Settings** page.

Map Source Metadata to Fields in a Content Type

The WebCenter Content v2.0 connector and some custom content connectors (connectors with type METADATA) allow you to map source metadata to Oracle Content Management content type fields.

Typically a source system has metadata associated with each content file. When you import a content file to an asset repository, you can use that source metadata to populate the fields in a content item, such as an employee record.

You can have the connector automatically create a content type for the content imported through a connector, with each piece of source metadata automatically mapped to a field in the content type. You can also customize the automatic mappings or map your own content type to the source content type. If you want to use your own content type, you must create it before creating mappings in the connector.

If your source content repository has multiple content types, you can select a different Oracle Content Management v2.0 content type to map to each source type.

Content type and field mapping must be done for all source types before the content connector is enabled. If you want to change the mappings, you need to disable the connector, make your changes, and then enable the connector again. Whenever an administrator saves the connector after modifying a mapping, the connector framework will attempt to seed the content type again. If you remove any associated content type that is required for the selected connector, the save will fail.

To map source metadata to fields in a content type:

1. Sign in to Oracle Content Management as an Administrator or Developer.
2. Click **Integrations** in the **Administration** area of the navigation menu on the left.
3. In the **Integrations** menu, choose **Content Connectors**.
4. Click **Configure** next to the connector you want to add mappings to.
5. If you want to automatically create a content type for imported content, with content type fields that are automatically mapped to the source metadata, select **Allow content type creation**. You can change the automatic mappings on the **Mappings** tab.
6. To create manual mappings or to edit the automatic mappings, click the **Mappings** tab:
 - a. Select a **Source Content Type**, then select the Oracle Content Management v2.0 content type you want to map to it.

- b. Map each **Source Data Field** to a **Target Data Field**.
 - c. Repeat for each source content type.
7. When you're done, click **Save**.

Provide Configuration Parameter Values for a Content Connector

A service administrator can register a content connector on the cloud storage provider's website to get configuration parameter values to enter for the content connector in Oracle Content Management.

Only one set of user credentials is stored for a signed-in Oracle Content Management user. If you need to use a separate user to fetch data, use Manage Sources and revoke the authorization for the existing user. Then you can use credentials for a second user.

Before you configure the content connector in Oracle Content Management, you need to get custom field information from the third-party content repository's website (such as Google Developers) by registering the content connector as an application that you want to integrate.

Here's the information you'll need for the various content connectors:

- Google Drive: Client ID, Client Secret, Developer Key, and App Id
 - Microsoft OneDrive: Client ID and Client Secret
 - Dropbox: App Key and App Secret
 - Microsoft SharePoint Online: Client ID, Client Secret, Sharepoint URL, Tenant ID, and Access Scope
 - WebCenter Content: WebCenter Content Server JAX-WS Connection URI, WebCenter Content Server JAX-WS Client Policy, and WebCenter Content Server Web Context Root
1. Click **Integrations** in the Administration area of the navigation menu.
 2. In the Integrations menu, choose **Content Connectors**.
 3. Click the **Configure** button next to a content connector.
 4. Click the **Additional Fields** tab, and enter the information you collected from the third-party content repository's website.
 5. Click **Save** to save the configuration parameter values.

After a content connector is configured and enabled, it's available in the asset repository for Oracle Content Management users to download content, through the **Add From** drop-down menu on the **Assets** page.

Delete a Content Connector

A service administrator can delete a disabled custom content connector in an Oracle Content Management instance at any time.

You cannot delete a preconfigured content connector. You can delete only newly added content connectors (custom content connectors).

To delete a content connector:

1. Click **Integrations** in the Administration area of the navigation menu.

2. In the **Integrations** menu, choose **Content Connectors**.
3. If the content connector is enabled, click **Disable** next to it.
4. Click **Delete** next to the custom content connector.

Part III

Developing Oracle Content Management Extensions

This part provides information on developing Oracle Content Management extensions. It includes the following chapters:

- [Develop Custom Actions with Application Integration Framework \(AIF\)](#)
- [Develop Content Connectors](#)
- [Develop Custom Field Editors](#)
- [Develop Custom Content Forms](#)
- [Develop Translation Connectors for Language Service Providers](#)
- [Develop External Processors](#)
- [Compile Content Layouts as HTML](#)

5

Develop Custom Actions with Application Integration Framework (AIF)

You can use Application Integration Framework (AIF) to define actions that are exposed in the web interface, respond to user selections, call third-party services, and specify how the results are presented to the user.

- [Understand the Application Integration Framework \(AIF\)](#)
- [Manage Custom Applications](#)
- [Configuration File Format](#)
- [Application Properties](#)
- [Action Command](#)
- [Invoke Command](#)
- [Presentation Command](#)
- [Expressions](#)
- [Variables](#)
- [Localization](#)

Understand the Application Integration Framework (AIF)

Application Integration Framework (AIF) provides a simple and effective way to integrate third-party services and applications into the Oracle Content Management web interface.

Using AIF, you can quickly define the actions that are exposed in the web interface, respond to user selections, call third-party services, and specify how the results are presented to the user. The framework supports variables and expressions and provides multiple language support.

Custom AIF applications are not applied when you access them through an applink or public link.

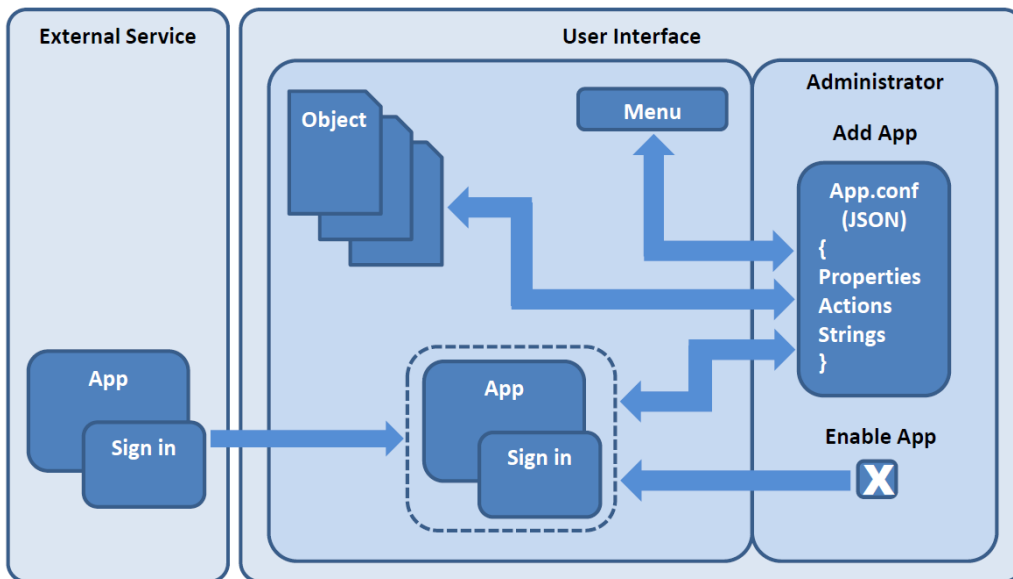
The definition for one or more integrations is stored in a single file in JSON format. As a developer, you can upload the configuration file and add it to a list of available applications. You can also edit and validate the configuration file directly in the web interface, enable or disable the app for general use, set preferences such as providing tenant and account information, download the configuration file, or delete the app.

Note:

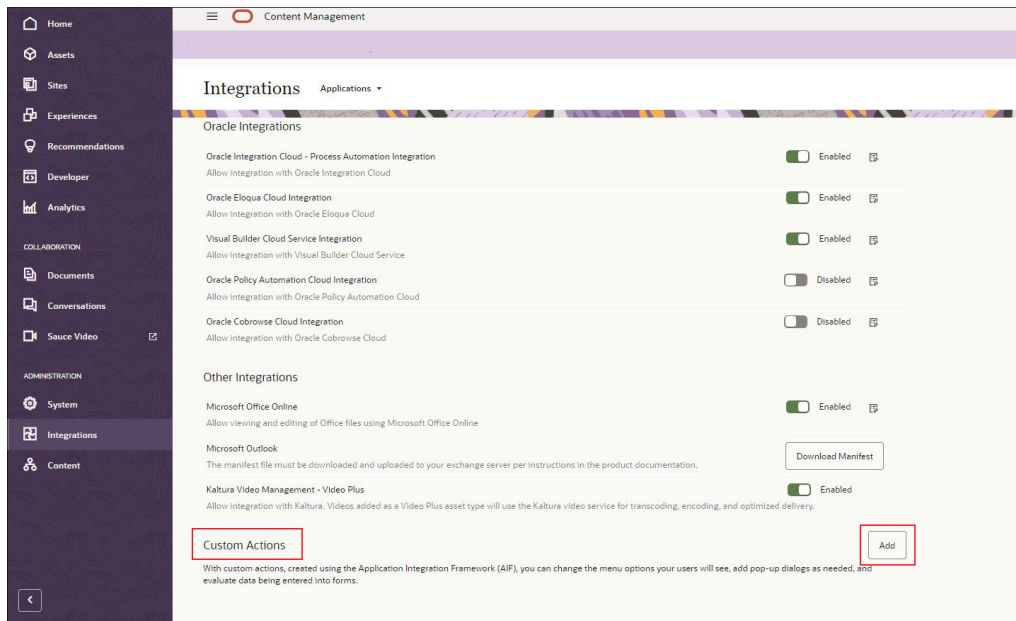
When using a private instance or any network environment with limited access, you'll need to provide users access to any third-party locations so they can use any custom actions.

The configuration file defines and manages the interactions between the app, native objects and web interface elements. The configuration file includes:

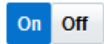






- App properties including tenant and user preferences
- Actions that are exposed in the web interface and the service calls they make
- How the results are presented to the user
- Interface strings with support for multiple languages



To manage apps created with Application Integration Framework, sign in as a developer, open your user menu, choose **Administration**, and then choose **Integrations**. Under Custom Actions, click **Add**.



From the **Applications** page, you can use the following options.

Setting	Description
	Enable or disable the application for users. When you enable the application, you can specify preferences for the application from the user menu, by choosing Preferences and then Applications Settings . You specify the user preferences resource in the <code>userPrefs</code> element in the configuration file.
	Browse local folders and files to locate and upload an application configuration file.
	Display the information defined for the application and specified in the <code>info</code> element of the configuration file.
	Display the preferences resource defined in the <code>tenantPrefs</code> element of the configuration file.
	Open the configuration file in the integrated JSON editor. The editor validates the syntax of the file to ensure that the file contains valid JSON code. Changes you make to the configuration file are immediately available in the enabled application. Changes you make to the configuration file are stored only in the server copy of the file. To back up your changes, use the Download icon to save the file locally.
	Download the file from the server to a local destination.
	Delete the configuration permanently. When you delete a configuration file, the deletion is permanent. The file can't be restored from the trash.

Manage Custom Applications

You can create custom applications using the Application Integration Framework (AIF).

With custom applications, you can change the menu options your users will see, add pop-up dialogs as needed, call third-party services, and specify how the results are presented to the user.

For details about creating applications using AIF, see [Understand the Application Integration Framework \(AIF\)](#).

After creating an application, you can add it to Oracle Content Management and manage it from the Administration web interface.

1. After you sign in to the Oracle Content Management web application as a service administrator, click **System** in the Administration area of the navigation menu.
2. In the **Settings** menu, click **Integrations**.
3. To add an application, under **Custom Actions**, click **Add** and navigate to the location where the configuration file that contains the application information is stored and select it.
4. After adding the application, you can enable or disable it by selecting or clearing the check box.

After you add your application and enable it, you can manage it by clicking the appropriate icon for any of the following actions:

- View information about the application. What is shown is set by the application's `info` application property (for example, a popup window).
- Set the preferences for the application. What is shown is set by the application's `tenantPrefs` property.
- Edit the application. You can alter the application's code and click **Load** to test the application, save the application, or cancel your edit. If an action isn't allowed in the code, an error message is displayed, describing the error.
- Download the application. You can save the application or open it using an editor of your choice.
- Remove the application. When you delete an application, it is not moved to your trash. You will need to add the application file again if you want to use it.

When an application is disabled, you can only edit the application, download it, or delete it.

To view all enabled applications, open your user menu, click **Preferences**, and then, in the Preferences menu, select **Applications**. You can view the information for the application and the preferences. The **Applications** option is not shown in the Preferences menu unless at least one application is enabled.

Configuration File Format

The definition for one or more integrations is stored in a single configuration file in JSON format.

The extension of an AIF configuration file must be `.conf`.

As an developer, you can upload the configuration file and make it available to users. See [Understand the Application Integration Framework \(AIF\)](#).



Note:

When using a private instance or any network environment with limited access, you'll need to provide users access to any third-party locations so they can use any custom actions.

A sample configuration file for one application follows. All keys and values in the configuration are case-sensitive.

```
{
  "id": "ExampleCoPublishing",
  "name": "APP_SUBMIT_TO_PUBLISH_DOCS_NAME",
  "description": "APP_SUBMIT_TO_PUBLISH_DOCS_DESC",
  "category": "CUSTOM",
  "supportEmail": "support@example.com",
  "baseUrl": "http://www.example.com/",
  "info": {
    "documentation": "Opens a URL with a description of the app in a popup window",
  },
  "info": {
    "presentation": {
      "view": "POPUP"
    }
  },
}
```

```
"invoke": {
  "method": "GET",
  "url": "http://www.example.com/ExampleCoDescr.jsp",
  "data": ""
},
},
"tenantPrefs": {
  "presentation": {
    "view": "POPUP"
  },
  "invoke": {
    "method": "GET",
    "url": "http://www.example.com/ExampleCoAdmin.jsp",
    "data": ""
  }
},
"userPrefs": {
  "presentation": {
    "view": "POPUP"
  },
  "invoke": {
    "method": "GET",
    "url": "http://www.example.com/ExampleCoUser.jsp?user={user.id}",
    "data": ""
  }
},
"actions": [
  {
    "id": "submitToPublish",
    "name": "ACTION_SUBMIT_TO_PUBLISH_NAME",
    "description": "ACTION_SUBMIT_TO_PUBLISH_DESC",
    "type": "UI",
    "trigger": "MENU",
    "presentation": {
      "view": "POPUP",
      "popupWidth": 700,
      "popupHeight": 400
    },
    "evaluate": "type=='folder' && user.isMember && user.role=='owner' && !
isReservedByAnotherUser",
    "invoke": {
      "method": "GET",
      "url": "http://www.example.com/ExampleCoSubmit.jsp?
user={user.id}&ids=[{id},]&names=[{name},]"
    },
    "multi": false
  },
  {
    "id": "showPublishStatus",
    "name": "ACTION_SHOW_PUBLISH_STATUS_NAME",
    "description": "ACTION_SHOW_PUBLISH_STATUS_DESC",
    "type": "UI",
    "trigger": "SELECT",
    "presentation": {
      "view": "POPUP",
      "popupWidth": 300,
      "popupHeight": 200
    },
    "evaluate": "type=='file' && user.isMember && user.role=='owner' &&
_.contains(['doc', '', 'docx', 'xls', 'xlsx', 'ppt', 'pptx', 'tif', 'png'], extension) &&
(_.isEmpty(reservedById) || reservedById === user.id)",
```

```

        "invoke": {
            "method": "GET",
            "url": "http://www.example.com/ExampleCoSubmit.jsp?
user={user.id}&ids=[{id},]&names=[{name},]"
        },
        "multi": false
    },
    {
        "id": "publishDirect",
        "name": "ACTION_PUBLISH_DIRECT_NAME",
        "description": "ACTION_PUBLISH_DIRECT_DESC",
        "type": "DIRECT",
        "trigger": "MENU",
        "presentation": {
            "view": "CLIENT",
            "popupWidth": 500,
            "popupHeight": 100
        },
        "evaluate": "type=='folder' && user.isMember && user.hasPrivilegesAs(item, 'owner')",
        "invoke": {
            "method": "GET",
            "url": "http://www.example.com/ExampleCoMenuAction.jsp"
        },
        "multi": false
    }
],
"stringsMap": {
    "en": {
        "APP_SUBMIT_TO_PUBLISH_DOCS_NAME": "ExampleCo Publishing Technical Articles",
        "APP_SUBMIT_TO_PUBLISH_DOCS_DESC": "Submit an article for the ExampleCo technical
knowledge base.",
        "ACTION_SUBMIT_TO_PUBLISH_NAME": "Publish",
        "ACTION_SUBMIT_TO_PUBLISH_DESC": "Submits the selected document for review and
approval to ExampleCo publishing.",
        "ACTION_SHOW_PUBLISH_STATUS_NAME": "Select ExampleCo",
        "ACTION_SHOW_PUBLISH_STATUS_DESC": "Shows currently selected row info in ExampleCo
Publishing",
        "ACTION_PUBLISH_DIRECT_NAME": "Audit",
        "ACTION_PUBLISH_DIRECT_DESC": "Sends an audit event to ExampleCo when document is
selected"
    }
}
}

```

The Application Integration Framework supports simple use of the JavaScript underscore library (<http://underscorejs.org>); however, AIF does not allow defining functions as parameters of underscore functions.

The sections that follow describe the functional areas of the Application Integration Framework.

Functional Area	Description
Application Properties	Describes the application properties.
Action Command	Describes the general elements of the <code>action</code> command. Use the <code>action</code> command to specify how to trigger and start a task. The <code>invoke</code> and <code>presentation</code> components of the <code>action</code> command are presented in their own sections.

Functional Area	Description
Invoke Command	Describes the elements of the <code>invoke</code> command. Use the <code>invoke</code> command to call a third-party service, page, or script.
Presentation Command	Describes the elements of the <code>presentation</code> command. Use the <code>presentation</code> command to specify how the third-party content retrieved by the <code>invoke</code> command is presented to the user.
Expressions	Provides examples and general information about expressions used with the <code>invoke</code> and <code>evaluate</code> commands.
Variables	Provides descriptions of the available item and user variables, the <code>permissions</code> and <code>status</code> objects, and AIF functions.
Localization	Provides an example and general information about how to localize an app with translated labels and descriptions.

Application Properties

The following table shows the properties of an Application Integration Framework application.

Property	Required	Description
<code>id</code>	Yes	A non-empty string that uniquely identifies the application across Oracle Content Management.
<code>name</code>	Yes	A non-empty string that is displayed for the application's web user interface. It can also be a key that can be translated using translation maps defined in <code>stringsMap</code> .
<code>description</code>	No	A string that is displayed with the application name in the web user interface. It can also be a key that can be translated using translation maps defined in <code>stringsMap</code> .
<code>category</code>	No	Category of the application. <code>CUSTOM</code> is the only allowed value. If you do not explicitly include <code>category</code> in the configuration file, it is dynamically added during application configuration and is given the default value of <code>CUSTOM</code> .
<code>supportEmail</code>	No	Email address of application support owner.
<code>baseURL</code>	No	Base URL used as a prefix to any relative URLs specified in the application.
<code>stringsMap</code>	No	Used to map languages to translation objects, so that keys in the application definition can be translated to the corresponding language.
<code>info</code>	No	Allows an integrator to show a dialog to display information about the application. It contains presentation and invoke properties.
<code>documentation</code>	No	Lets you add comments in the file, such as a comment to identify the purpose of an action. You might add a <code>documentation</code> property at the start of the code sample to describe the application itself.
<code>tenantPrefs</code>	No	Allows an integrator to expose application-level options in the preference settings. It contains presentation and invoke properties.

Property	Required	Description
userPrefs	No	Allows an integrator to expose application-level options in the preference settings for users. It contains presentation and invoke properties.
actions	Yes	A collection of actions contained in the application. It must contain at least one action .

Action Command

Use the `action` command to specify how to trigger and start a specified task and how to present the results.

The following table shows the properties of an `action` command.

Property	Required	Description
id	Yes	A non-empty string that uniquely identifies the action in the application.
name	Yes	A non-empty string that is displayed for the action in the web user interface. It can also be a key that can be translated using translation maps defined in <code>stringsMap</code> .
description	No	A string that is displayed for the action in a web user interface tooltip. It can also be a key that can be translated using translation maps defined in <code>stringsMap</code> .
trigger	No	Specifies if the action is triggered through a context menu taskbar (<code>MENU</code>), or when the item is <i>opened</i> (<code>SELECT</code>). If you do not explicitly include <code>trigger</code> in the configuration file, it is dynamically added during application configuration and is given the default value of <code>MENU</code> . In the Application Integration Framework, the <code>SELECT</code> trigger occurs <i>when</i> an <i>item</i> is opened, not when simply checked in a folder listing.
type	No	Specifies how action results are displayed to the end user. Possible values are <code>UI</code> and <code>DIRECT</code> . If the value is <code>DIRECT</code> , a call is made to the URL specified with <code>invoke</code> , and the response displays as a notification in the Oracle Content Management web interface, if possible. If the value is <code>UI</code> , the <code>presentation</code> property specifies the type of web interface to use.
presentation	No	Specifies the presentation to use if <code>type</code> is set to <code>UI</code> . Possible values are <code>CLIENT</code> to embed the action results in the integrations side panel on the right of the display, <code>POPUP</code> to display action results in a separate browser dialog, and <code>WINDOW</code> to display action results in a new browser window or tab. If <code>type</code> is set to <code>UI</code> and <code>presentation</code> is omitted, it is added to the action with the default value of <code>CLIENT</code> . See Presentation Command .

Property	Required	Description
multi	No	Specifies if the action applies when multiple items are selected (<code>true</code>) or not (<code>false</code>). If you do not explicitly include <code>multi</code> in the configuration file, it is dynamically added during application configuration and is given the default value of <code>false</code> .
evaluate	Yes	Must be valid JavaScript Boolean value that evaluates to <code>true</code> or <code>false</code> for the selected item. For more information about expression evaluation, see Expressions .
invoke	Yes	Specifies the external service in an <code>invoke</code> command to use when action is triggered. See Invoke Command .

Invoke Command

Use the `invoke` command to call an external service, page, or script.

The following table shows the properties of the `invoke` command.

Property	Required	Description
method	Yes	Specifies the HTTP method to use for invocation. Valid values are <code>GET</code> , <code>POST</code> , <code>PUT</code> , <code>DELETE</code> , and <code>HEAD</code> .
url	Yes	A URL expression to specify the URL for the third-party service. It can be an absolute or relative URL. If a relative URL is specified, it will be added to the <code>baseUrl</code> of the application, if specified.
data	No	An expression that specifies data to be sent with the URL. See Expressions . A JSON payload can added to the request body as a string object.
appLinkRole	No	If the action is a request to create an application link (<code>applink</code>), use <code>appLinkRole</code> to specify the role to use for the <code>applink</code> . Valid values are <code>contributor</code> , <code>downloader</code> , and <code>viewer</code> .
header	No	A custom header to include when making POST calls.
token	No	Specifies if any special security token needs to be passed. Valid values are <code>PCS</code> (PCS OAUTH token) and <code>NONE</code> .

Presentation Command

Use the `presentation` command to specify how the third-party content retrieved by the `invoke` command is presented to the user.

The following table shows the properties of the `presentation` command.

Property	Required	Description
view	No	Specifies how action response content is displayed to the user. Use <code>CLIENT</code> to embed results in the application, <code>POPUP</code> to display results in a separate browser dialog, or <code>WINDOW</code> to display results in a new browser window or tab. If you use <code>CLIENT</code> with the <code>SELECT</code> trigger, action results are displayed in the integrations side panel on the right of the display. The integrations panel must be open for the results to display. The <code>SELECT</code> trigger occurs when an item is opened, not when simply checked in a folder listing. If you use <code>CLIENT</code> with the <code>MENU</code> trigger, action results are displayed in a pop-up dialog. If you do not explicitly include <code>view</code> in the configuration file, it is dynamically added during application configuration and is given the default value of <code>CLIENT</code> .
popupWidth	No	A number specifying the width of the container. If none is specified, a default value is used, depending on the context.
popupHeight	No	A number specifying the height of the container. If none is specified, a default value is used, depending on the context.

Expressions

The `invoke` and `evaluate` commands accept expressions, which can include variables and operators that are evaluated at runtime.

Invoke Expressions

An *invoke expression* is a template used to generate URL and data values at runtime by replacing the variables specified in the template with their runtime values. Variables can reference one or more selected items, information about the current user, and other system information.

- Anything enclosed in a brace `{ }` is considered to be a key value that is mapped to the corresponding attribute of a file, folder, or the currently signed-in user. User attributes are prefixed by `user`.

For example, if John Smith is the current user, `userid={user.name}` resolves to this:

```
userid=John Smith
```

- For multiselection, use the *[repeated template separator]* syntax. The repeated template is resolved for each selected item, and resolved strings for items are separated by the specified delimiter.

For example, for items with GUIDs `x189` and `y234`, `item={{id},}` resolves to this:

```
item=x189,y234
```

Evaluate Expressions

An *evaluate expression* determines if the item or items selected by the user qualify for a particular action. The expression can contain zero or more variables that are

replaced with their runtime values before the expression is evaluated. This expression must be a valid JavaScript expression after all the values of the variables are replaced with their values at runtime.

The expression must evaluate to `true` or `false`. If the result is `true`, the action is made available as specified by the `trigger` property. If the result is `false`, the action is not made available to the user.

For example, the following expression returns a `true` value only if all three conditions evaluate to `true`: the item selected in the interface is a file, the user is signed in (not accessing the file using a public link or `aplink`), and the user is the owner of the file.

```
"evaluate": "type=='file' && user.isMember && user.role=='owner'"
```

Another example is `item.meta.<custom property group name>.<custom property group field name>`.

Some commonly used JavaScript comparison and logical operators follow.

Operator	Description
<code>==</code>	equal to
<code>!=</code>	not equal to
<code>></code>	greater than
<code><</code>	less than
<code>=</code>	greater than or equal to
<code><=</code>	less than or equal to
<code>&&</code>	logical and
<code> </code>	logical or
<code>!</code>	logical not

Variables

You can use variables in applications to dynamically provide information about the current user, the currently selected item or items, and other types of information. Variables are evaluated at runtime when the application is used.

You can use a category prefix (`item` or `user`) with variable names to make the relationship explicit, especially in cases where variable names are the same, such as `name`. If a variable is not prefixed by any category, `item` is used as the category.

For example, to specify the folder or file name, use `item.name`. To specify the user name, use `user.name`. If you specify `name` without the category, then `item.name` (the folder or file name) is assumed.

Folder and File Item Variables

The following variables are specific to a folder or file item that is currently selected by the user. Folder and file variables belong to the `item` category. The following table lists variables that apply to both folder and file items.

Folder or File Variable	Description
id	Item GUID
name	Item name
description	Item description
type	Item type
parentid	Item parent ID
ownerid	Item owner GUID
ownername	Item owner name
creatorname	Item creator name
createdat	Date and time item was created
lastmodifierid	GUID of the user who most recently modified the item
lastmodifiername	Name of the user who most recently modified the item
modifiedat	Date and time the item was last modified
size	Size of the item
permissions	Actions the user is allowed to perform on the item
state	State of the item
applink	Applink created by AIF for this user to access the item, if <code>appLinkRole</code> was specified in the <code>invoke</code> command The <code>appLink</code> object has <code>id</code> , <code>url</code> , and <code>accessToken</code> properties that can be passed in the URL or data as <code>{appLink.id}</code> , <code>{appLink.url}</code> , and <code>{appLink.accessToken}</code> , respectively.
favorite	Whether the item is a favorite of the user or not

File-Item Specific Variables

The following variables are specific to a file item that is currently selected by the user. File variables belong to the `item` category. The table lists variables that apply to file items only.

File Variable	Description
originalname	Original name of the item
extension	Extension of the item
revision	Item's revision
mimetype	Mime type of the item
reservedbyid	GUID of the user who reserved the item
reservedbyname	Name of the user who reserved the item
reservedat	Date and time when the item was reserved
isreservedbyanotheruser	Whether the item is reserved by a user other than the current user

Item Permissions

`Permissions` is a JSON object that you can query to determine the permissions associated with an item. The properties return a Boolean value.

Permission Properties	Description
annotationDelete	User has delete permission for annotations on the item.
annotationRead	User has read permission for annotations on the item.
annotationUpdate	User has update permission for annotations on the item.
annotationWrite	User has write permission for annotations on the item.
directShareDelete	User has delete permission for the item shared with the user.
directShareRead	User has read permission for the item shared with the user.
directShareUpdate	User has update permission for the item shared with the user.
directShareWrite	User has write permission for the item shared with user.
fileDelete	User has delete permission for the file.
filePreview	User has preview permission for the file.
fileRead	User has read permission for the file.
fileUpdate	User has update permission for the file.
fileWrite	User has write permission for the file.
folderDelete	User has delete permission for the folder.
folderPreview	User has preview permission for the folder.
folderRead	User has read permission for the folder.
folderUpdate	User has update permission for the folder.
folderWrite	User has write permission for the folder.
linkShareDelete	User has delete permission for the item shared with the user through a link.
linkShareRead	User has read permission for the item shared with the user through a link.
linkShareUpdate	User has update permission for the item shared with the user through a link.
linkShareWrite	User has write permission for the item shared with the user through a link.

Item State

`state` is a JSON object that you can query to determine certain states of an item. The properties return a Boolean value.

State Property	Description
isAnnotated	The item has annotations.
isAnnotatedLatest	The item has the most recent annotations.
isLinked	The item has named sharing links defined for it.
isShared	The item is directly shared with specified users.
isSyncd	The item is included in the content synced through the desktop client.
isInTrash	The item is currently in the trash.

User Variables

The following variables are relative to the current user or session, or both.

Variable	Description
id	User's GUID
name	User's name
loginname	User's sign in name
email	User's email address
timezone	User's time zone
ismember	Returns <code>true</code> if the user is not accessing an item through public link or app link and <code>false</code> otherwise
ispubliclink	Returns <code>true</code> if the user is accessing the item through a public link and <code>false</code> otherwise
isapplink	Returns <code>true</code> if the user is accessing the item through an applink and <code>false</code> otherwise
role	Current user's role for an item
hasprivilegesas	Specified with a user role as a string argument to determine if the user has the specified role

API Functions

Use the following utility functions with `evaluate` or `invoke` expressions to return information about a specified item.

Variable	Description
<code>isReservedByAnotherUser(item)</code>	Returns <code>true</code> if the item is reserved by another user, or returns <code>false</code> otherwise
<code>hasPrivilegesAs(item, role)</code>	Returns <code>true</code> if the user has role privileges for the item, or returns <code>false</code> otherwise
<code>getUserRole(item)</code>	Returns the user's role for the item

Localization

Use the `stringsMap` element to provide localized versions of labels and descriptions used in the integration.

The following example shows sample application and action names and descriptions in English and German.

```

"stringsMap": {
  "en": {
    "APPLICATION_NAME": "Localization Test",
    "APPLICATION_DESCRIPTION": "This example provides translated
content.",
    "ACTION1": "Action1",
    "ACTION2": "Action2",
    "ACTION1DESC": "Action1 description",
    "ACTION2DESC": "Action2 description"
  },
  "de": {
    "APPLICATION_NAME": "Lokalisierung Beispiel",
    "APPLICATION_DESCRIPTION": "Dieses Beispiel liefert übersetzten

```

```
Inhalte.",  
    "ACTION1": "Aktion1",  
    "ACTION2": "Aktion2",  
    "ACTION1DESC": "Aktion1 Beschreibung",  
    "ACTION2DESC": "Aktion2 Beschreibung"  
}
```

6

Develop Content Connectors

You can use the Oracle Content Management connector framework to develop your own content connectors to bring content you have already created elsewhere into Oracle Content Management, manage it centrally, and use it in new experiences across multiple channels.

Part of Oracle Content Management, the connector framework integrates with remote content stores to let you bring content into Oracle Content Management as easily as uploading files from your desktop. You can plug in any type of content store to Oracle Content Management, making it a content hub.

The connector framework abstracts the functionality that is provided and implemented by end connectors. It does so by providing the REST API interface to configure a content connector as well as to call content-related APIs on the end store, like browsing the remote file system and copying the files from a remote store to Oracle Content Management.

The following sections describe how to develop custom content connectors:

- [Connector REST API Interface](#)
- [Connector SDK](#)
- [Build a New Content Connector](#)
- [Content Connector Configuration and Registration](#)
- [Content Connector Execution Flow](#)
- [Pexels Content Connector Sample Implementation](#)
- [Download the CEC Content Connector Sample and SDK](#)

Connector REST API Interface

You can deploy and run content connectors anywhere and implement them with any technology stack, as long as they can be called from Oracle Content Management through REST APIs.

Content connectors are deployable binaries that use RESTful services for the remote stores implementing the REST interfaces (SPIs/SDK) defined by Oracle Content Management. They internally use cloud store native SDKs to connect to remote systems.

Connector SDK

To help you build a new content connector, Oracle Content Management provides the Connector SDK

This SDK contains:

- A javadoc
- A JAR file containing DTO and REST interfaces
- The `connector.yaml` file, which documents the REST APIs and the payloads

The Connector SDK is in the `cec-connector-sdk.zip` file.

Build a New Content Connector

To build a new content connector as a RESTful service, a developer needs to implement the RESTful interfaces provided by the connector framework.

The Oracle Content Management connector framework has the following interfaces.

- `APIResource` Interface: This interface is the starting point of the content connector. You simply return the version that's supported.
- `ServerResource` Interface: When a developer or administrator registers a content connector in the Oracle Content Management server, Oracle Content Management calls this service to get the basic details about the connector server, such as these:
 - Authorization type:
 - * `OAuth`: If the remote cloud store supports three-legged OAuth authentication
 - * `BASIC_AUTH`: If the remote cloud store supports Basic authentication

 **Note:**

Oracle no longer supports Basic authentication for general consumption.

- * `NO_AUTH`: If the remote store doesn't require any authentication
- Picker type: Oracle Content Management supports the Oracle Content Management generic picker to list the file system. If the content connector supports the native picker, then the content connector implementer can implement JavaScript APIs to integrate that picker with the Oracle Content Management UI. The picker type can be one of three values:
 1. `COMMON`, using the common UI
 2. `CUSTOM`, using a custom UI
 3. `NATIVE`, using a native UI like Google Drive or Microsoft OneDrive
- Also use this Interface to define custom fields for the content connector. Custom properties are connector specific, so the connector framework can't provide such a properties list by itself. Every content connector has its own requirement to connect to a remote store; for example, one content connector might need just `ClientID` and `ClientSecret`, while another might require `ClientID`, `ClientSecret`, `AppID`, and so on. Each content connector can provide a custom properties list to the Oracle Content Management server by `ServerResource`. If any of these properties need to be filled in by a developer or administrator during configuration, the connector framework will surface them in the administration UI.

This service will be called only once, at the time of registration.

- `AuthorizationResource` Interface: Used to complete the authorization before content from the remote store can be copied to the Oracle Content Management

repository. This interface has support for both OAuth and Basic authentication protocols.

- Three-legged OAuth authentication
 - * In the OAuth flow, the Oracle Content Management UI, based on the custom property value provided by the content connector, supports OAuth authentication. It triggers an OAuth flow by calling the connector framework, which in turn calls the `/authorization/authorization` URLs service on the content connector to get the authorization URL.
 - * The Oracle Content Management UI renders the authorization URL in a browser, where the user authorizes the application and gives Oracle Content Management permission to access the token for further communication. To complete the authorization and get the OAuth token, the Oracle Content Management server will call the `/authorization/completedAuthorizations` service on the content connector. To complete the three-legged OAuth, the Oracle Content Management server also provides the redirect callback servlet, which the remote store will call to complete the authorization process.
- Basic authentication: In case the content connector supports Basic authentication, the Oracle Content Management UI will prompt the user to enter a user name and password, which it will pass to the Oracle Content Management (connector framework). The connector framework will call the `/authorization/basicAuthorization` service to pass the credentials and will expect the content connector to validate with the remote store if the credentials are correct.
- `FileSystemResource` Interface: The connector framework supports the generic file picker, which can be used by content connectors to render the file-system information for a user to browse the files. But the content connector needs to implement the `/filesystem` service to use this functionality. Highlights of this service are the following:
 - The user can browse the file system like any other file picker; for example, the Google Drive or Microsoft OneDrive picker.
 - This supports pagination.
 - The user can also specify the search criteria to get the filtered result.
- `ContentResource` Interface: The connector framework uses the service `/content` to get the actual content (the bytes of the files) stored in the cloud store.

This service returns `javax.ws.rs.core.Response` as a response, and the response entity is the content of the file. The response should include the following headers:

- `Content-Length`: Set to the entity length, or `-1` if unknown.
- `Content-Type`: Set to the content type, or `application/octet-stream` if a more accurate type is not possible.
- `Content-Disposition`: Set to the disposition type of attachment, and includes a `filename` parameter (for example, `Content-Disposition: attachment; filename="meeting agenda.doc"`). Note that file names with spaces must be in quotation marks. See RFC 6266 for details about the format and encoding of the header.

Each of these interfaces is described in more detail, with examples of REST payloads, in the following sections:

- [REST Interfaces for Configuration, Authorization, and Fetching Content](#)
- [REST Interfaces for File System Browsing and Searching](#)

- [Content Picker](#)
- [Authorization](#)

REST Interfaces for Configuration, Authorization, and Fetching Content

A content connector needs to implement the following REST APIs for defining the connector configuration, setting up authorization, and fetching content.

/rest/api

Implements `intradoc.connectorcommon.server.APIResource`

Here you return the latest version supported by the content connector.

GET `http://host:port/connector/rest/api`

```
[ "v1" ]
```

/rest/api/v1/server

Implements `intradoc.connectorcommon.server.ServerResource`

This returns information about the content connector configuration, like the authentication type, picker type, and custom fields it exposes.

GET `http://host:port/connector/rest/api/v1/server`

```
{
  "name": "Pexels Connector",
  "nameLocalizations": [
    {
      "locale": "en",
      "localizedString": "Pexels Connector"
    }
  ],
  "version": " ( , , )",
  "about": "Pexels Connector.<br>Copyright (c) 2019, Oracle and/or
its affiliates. All rights reserved.",
  "aboutLocalizations": [
    {
      "locale": "en",
      "localizedString": "Pexels Connector.<br>Copyright (c)
2019, Oracle and/or its affiliates. All rights reserved."
    }
  ],
  "authenticationType": "NO_AUTH",
  "pickerType": "CUSTOM",
  "enableMultiUserCopyBack": false,
  "maxUploadSize": 1073741824,
  "fields": [
    {
      "ID": "ProxyHost",
      "datatype": "STRING",
```

```
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "HTTP Proxy Hostname",
        "labelLocalizations": [
            {
                "locale": "en",
                "localizedString": "HTTP Proxy Hostname"
            }
        ],
        "description": "The HTTP proxy hostname, leave blank to
disable.",
        "descriptionLocalizations": [
            {
                "locale": "en",
                "localizedString": "The HTTP proxy hostname, leave blank
to disable."
            }
        ],
        "required": false
    },
    {
        "ID": "ProxyPort",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "HTTP Proxy Port",
        "labelLocalizations": [
            {
                "locale": "en",
                "localizedString": "HTTP Proxy Port"
            }
        ],
        "description": "The HTTP proxy port number, leave blank to
default to port 80.",
        "descriptionLocalizations": [
            {
                "locale": "en",
                "localizedString": "The HTTP proxy port number, leave
blank to default to port 80."
            }
        ],
        "required": false
    },
    {
        "ID": "ProxyScheme",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "HTTP Proxy Scheme",
```

```

        "labelLocalizations": [
            {
                "locale": "en",
                "localizedString": "HTTP Proxy Scheme"
            }
        ],
        "description": "The HTTP proxy scheme, leave blank to
default to http.",
        "descriptionLocalizations": [
            {
                "locale": "en",
                "localizedString": "The HTTP proxy scheme, leave
blank to default to http."
            }
        ],
        "required": false
    },
    {
        "ID": "ClientID",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "Client ID",
        "labelLocalizations": [
            {
                "locale": "en",
                "localizedString": "Client ID"
            }
        ],
        "description": null,
        "descriptionLocalizations": [],
        "required": true
    }
],
"supportedConnectorTypes": [
    "COPY"
],
"proprietorName": "",
"serviceProviderName": "Pexels",
"nativeAppInfos": null
}

```

/rest/api/v1/authorization/authorizationURLs

Implements `intradoc.connectorcommon.server.AuthorizationResource`

This is required only if the content connector supports OAuth.

It returns the authorization URL to which the browser will redirect to invoke the OAuth flow where the user provides credentials and authorizes the access. The redirect URL

passed in the payload is what the OAuth provider will redirect to with the temporary code.

POST `http://host:port/connector/rest/api/v1/authorization/authorizationURLs`

Headers

```
Content-Type:application/json
X-CEC-ClientID:client-id
X-CEC-ClientSecret:client-secret
X-CEC-ProxyHost:proxy-host
X-CEC-ProxyPort:80
X-CEC-ProxyScheme:http
```

Payload

```
{"redirectURL":"http://host:port/documents/web/AR_COMPLETE_AUTHORIZATION"}
```

Response

```
{
  "authorizationURL": "https://domain/oauth/authorize?
  response_type=code&client_id=id://host:port/documents/web/
  AR_COMPLETE_AUTHORIZATION",
  "fieldValueMap": null
}
```

/rest/api/v1/authorization/completedAuthorizations

Implements `intradoc.connectorcommon.server.AuthorizationResource`

This is required only if the content connector supports OAuth.

This is called to complete the second part of the OAuth flow, where the code obtained from the OAuth provider in the previous step is passed along with the client ID and secret to obtain the access token and refresh token along with expiry times. This information is then returned to Oracle Content Management, which stores it securely against the signed-in user.

POST `http://host:port/connector/rest/api/v1/authorization/completedAuthorizations`

Headers

```
Content-Type:application/json
X-CEC-ClientID:client-id
X-CEC-ClientSecret:client-secret
X-CEC-code:code
X-CEC-ProxyHost:proxy-host
X-CEC-ProxyPort:80
X-CEC-ProxyScheme:http
Content-Type:application/json
```

Payload

```
{"redirectURL":"http://host:port/documents/web/AR_COMPLETE_AUTHORIZATION"}
```

Response

```
{
  "authorized": true,
  "authorizedUserDisplayName": null,
  "authorizedUserEmailAddress": null,
}
```

```
    "authorizedUserPictureURL": null,  
    "fieldValueMap": {  
        "RefreshToken": "refresh-token",  
        "AccessToken": "access-token"  
    }  
}
```

/rest/api/v1/authorization/basicAuthorization

Implements `intradoc.connectorcommon.server.AuthorizationResource`

This is required only if the content connector supports basic authorization. Here the sign in credentials are passed in the headers where the password field is base64 encoded. It is always recommended that a content connector be deployed on an SSL endpoint.

POST `http://host:port/connector/rest/api/v1/authorization/basicAuthorization`

POST `http://host:port/connector/rest/api/v1/authorization/basicAuthorization`

Headers

```
Content-Type:application/json  
X-CEC-UserName:user  
X-CEC-UserPwd:password  
X-CEC-ProxyHost:proxy-host  
X-CEC-ProxyPort:80  
X-CEC-ProxyScheme:http
```

Response

```
true
```

/rest/api/v1/content

Implements `intradoc.connectorcommon.server.ContentResource`

Given a file ID, this return the input stream for a file.

GET `http://host:port/connector/rest/api/v1/content?uri=fFileGUID:xxxx`

Request Headers

```
X-CEC-ClientID:client-id  
X-CEC-ProxyHost:proxy-host  
X-CEC-ProxyPort:80  
X-CEC-ProxyScheme:http
```

Response Headers

```
content-disposition attachment; filename=pexels-photo-xxxx.jpeg  
content-type image/jpeg
```

Response Body

```
File Content
```

REST Interfaces for File System Browsing and Searching

If a content connector uses the common picker available out of the box with Oracle Content Management, then the content connector needs to implement the following REST API.

This REST API is required for the common UI.

/api/v1/filesystem

Implements `intradoc.connectorcommon.server.FilesystemResource`

This interface supports search as well as where `queryText` is passed.

POST `http://host:port/connector/rest/api/v1/filesystem`

Request Headers

```
X-CEC-ClientID:client-id
X-CEC-ProxyHost:proxy
X-CEC-ProxyPort:80
X-CEC-ProxyScheme:http
Content-Type:application/json
```

Request Body

```
{
  "itemCount": "50",
  "itemStartRow": "0",
  "itemsSortField": "ASC",
  "queryText": "animals"
}
```

Response

```
{
  "numItems": 50,
  "hasMoreItems": true,
  "totalItemsCount": 7190,
  "fileSystemInfo": {
    "uri": "fFolderGUID:null",
    "parentUri": "fFolderGUID:null",
    "name": "/",
    "description": null,
    "isDirectory": true,
    "size": null,
    "mimeType": null,
    "extension": null,
    "creator": null,
    "createdTimeStamp": null,
    "lastModifiedBy": null,
    "lastModifiedTimeStamp": null,
    "browseURL": null,
    "thumbnailURL": null,
    "directoryContents": [
      {
        "uri": "45170",
        "parentUri": "fFolderGUID:null",
```

```
        "name": "kittens-cat-cat-puppy-rush-45170.jpeg",
        "description": "kitten cat rush lucky cat",
        "isDirectory": false,
        "size": null,
        "mimeType": "image/jpeg",
        "extension": "jpeg",
        "creator": "Pixabay",
        "createdTimeStamp": null,
        "lastModifiedBy": null,
        "lastModifiedTimeStamp": null,
        "browseURL": "https://images.pexels.com/photos/45170/
kittens-cat-cat-puppy-rush-45170.jpeg",
        "thumbnailURL": "https://images.pexels.com/photos/
45170/kittens-cat-cat-puppy-rush-45170.jpeg?
auto=compress&cs=tinysrgb&fit=crop&h=200&w=280",
        "directoryContents": null,
        "version": null,
        "additionalInformation": {
            "photographer": "Pixabay",
            "photographer_url": "http://api-server.pexels.com/
@pixabay"
        }
    },
    {
        "uri": "66898",
        "parentUri": "fFolderGUID:null",
        "name": "elephant-cub-tsavo-kenya-66898.jpeg",
        "description": "elephant cub kenya savanna",
        "isDirectory": false,
        "size": null,
        "mimeType": "image/jpeg",
        "extension": "jpeg",
        "creator": "Pixabay",
        "createdTimeStamp": null,
        "lastModifiedBy": null,
        "lastModifiedTimeStamp": null,
        "browseURL": "https://images.pexels.com/photos/66898/
elephant-cub-tsavo-kenya-66898.jpeg",
        "thumbnailURL": "https://images.pexels.com/photos/
66898/elephant-cub-tsavo-kenya-66898.jpeg?
auto=compress&cs=tinysrgb&fit=crop&h=200&w=280",
        "directoryContents": null,
        "version": null,
        "additionalInformation": {
            "photographer": "Pixabay",
            "photographer_url": "http://api-server.pexels.com/
@pixabay"
        }
    },
    . . .more entries. . .
    ],
    "version": null,
    "additionalInformation": null
}
```


The `itemCount` value represents the page size, and the `itemStartRow` value indicates the starting index of the current page requested. You can use these values to determine the page number to retrieve and the number of records to fetch from your backend.

Content Picker

Oracle Content Management supports two mechanisms for integration so users can browse and select content and assets to bring into Oracle Content Management.

The content picker for a content connector can be either the common content picker or a custom one:

- The common content picker is provided by Oracle Content Management.
With the generic common picker, Oracle Content Management provides a basic framework and capabilities to build content browsing and selection. This works seamlessly in conjunction with the Connector Framework SDK and the APIs defined.
- The custom picker needs to be built by the content connector and can be built using any JS technology.

The custom picker is typically used in cases where the content connector builds its own experience for accessing content in the remote store. Some stores have their own native pickers, which the content connector could reuse for integrating with the asset repository; for example, Google Drive, Microsoft OneDrive, and Dropbox. Additionally, the content connector can develop its own picker UI following the predefined interfaces and callback mechanisms prescribed by Oracle Content Management.

If the content connector uses the common picker, then the connector needs to define `pickerType` in the server implementation:

`intradoc.connectorcommon.server.ServerResource` implementation

```
serverInfo.pickerType = PickerType.COMMON;
```

The content connector also needs to implement the `intradoc.connectorcommon.server.FileSystemResource` interface. The implementation needs to adhere to following guidelines in its `filesystem` REST implementation:

1. The content connector must append "fFolderGUID:" in `uri`, `parentUri` for each folder in the response of `fileSystem`.
2. A few fields are mandatory for every folder and file to be set in `FileSystemResource` for the common UI to work without any issues: `uri`, `parentUri`, and `name`. If `parentUri` is not applicable, then it can be set to `fFolderGUID:null`.
3. You must know the size of the result set being returned up front; that is, `totalItemsCount` needs to be set.

Use the Common UI

If the content connector uses the common picker, then it needs to define `pickerType` in the `eserver` implementation:

`intradoc.connectorcommon.server.ServerResource` implementation

```
serverInfo.pickerType = PickerType.COMMON;
```

The content connector also needs to implement the `intradoc.connectorcommon.server.FileSystemResource` interface. The implementation needs to adhere to following guidelines in its `filesystem` REST implementation:

1. The content connector must append "fFolderGUID:" in `uri`, `parentUri` for each folder in the response of `filesystem`.
2. A few fields are mandatory for every folder and file to be set in `FileSystemResource` for the common UI to work without any issues: `uri`, `parentUri`, and `name`. If `parentUri` is not applicable, then it can be set to `fFolderGUID:null`.
3. You must know the size of the result set being returned up front; that is, `totalItemsCount` needs to be set.

The common UI supports customizations, which can be defined in the connector configuration. The following table describes the additional settings that can be passed in a connector's `ServerResource` implementation.

Property	Values	Description
<code>hide_breadcrumbs</code>	<code>true</code> <code>false (default)</code>	To show or hide breadcrumbs
<code>nextPrevPaginationEnabled</code>	<code>true</code> <code>false (default)</code>	To show page with next and prev buttons for pagination For numbered pagination, don't set this property to <code>true</code> .
<code>CommonPickerShowVideoView</code>	<code>true</code> <code>false (default)</code>	Card layout will be displayed, with each card having the following items: <ul style="list-style-type: none"> • Iframe with video link • Description of video • Channel name
<code>CommonPickerShowImageView</code>	<code>true</code> <code>false (default)</code>	Card layout will be displayed, with each card having the following items: <ul style="list-style-type: none"> • Image thumbnail with link to open full image in next tab • Photographer URL • Name of image • mimeType
<code>show_termsOfUse</code>	<code>true</code> <code>false (default)</code>	To show the terms and condition link
<code>termsOfUse_link</code>	String	If the <code>show_termsOfUse</code> property is <code>true</code> , then the link for terms and conditions is specified using this property.
<code>show_privacyPolicy</code>	<code>true</code> <code>false (default)</code>	To show the privacy policy link

Property	Values	Description
privacyPolicy_link	String	If the show_privacyPolicy property is true, then the link for the privacy policy is specified using this property.
show_view_action	true false (default)	To show the view action on select of item
page_size	String 50 (default)	The size of one page. If results are more than the given page size, then the results will be paginated.

The settings are in the form of a map that needs to be set on `ServerInfo`. An example follows:

```
//Add UI Settings for the connector that will be read by the common
picker
Map<String,Object> additionalSettingsMap = new HashMap<String,Object>();
Map<String,String> uiSettingsMap = new HashMap<String,String>();
uiSettingsMap.put("CommonPickerShowImageView", "true");
uiSettingsMap.put("hide_breadcrumbs", "true");
uiSettingsMap.put("show_termsOfUse","true");
uiSettingsMap.put("show_privacyPolicy","true");
uiSettingsMap.put("termsOfUse_link","https://unsplash.com/terms");
uiSettingsMap.put("privacyPolicy_link","https://unsplash.com/privacy");
uiSettingsMap.put("show_view_action","true");
additionalSettingsMap.put("UISettings", uiSettingsMap);
serverInfo.additionalSettings = additionalSettingsMap;
```

The resulting JSON structure follows:

```
"additionalSettings": {
  "UISettings": {
    "hide_breadcrumbs": "true",
    "show_view_action": "true",
    "CommonPickerShowImageView": "true",
    "termsOfUse_link": "https://unsplash.com/terms",
    "privacyPolicy_link": "https://unsplash.com/privacy",
    "show_privacyPolicy": "true",
    "show_termsOfUse": "true",
    "page_size": "30"
  }
}
```

Use a Custom UI

A content connector can implement a custom UI for its picker. This can be built using any JS technology. However, it needs to invoke some predefined methods in Oracle Content Management to obtain connector configuration and to pass back selections. The custom picker can be built to expose its own **OK** and **Cancel** buttons, in which case it needs to

invoke the appropriate handlers. If the picker doesn't expose its own **OK** and **Cancel** buttons, Oracle Content Management will provide this.

The `pickerType` needs to be defined as `CUSTOM` in the server implementation:

intradoc.connectorcommon.server.ServerResource implementation

```
serverInfo.pickerType = PickerType.CUSTOM;
```

The custom implementation needs to include the following JS file in its library:

oracle-oce-custompicker.js

```
/*global window:true*/
(function() {
  'use strict';

  var name = window.name; // name set on iframe element to uniquely
  identify the instance
  var receiver = window.opener || window.parent;
  var origin = "*";
  function _postMessage(msg) {
    msg.name = name;
    receiver.postMessage(msg, origin);
  }

  var namespace = "OCE"; //todo: allow passing in data attribute
  if (!window[namespace]) {
    window[namespace] = {};
  }
  var OCE = window[namespace];

  if (!OCE.CustomPicker) {

    OCE.CustomPicker = {

      selection: [],

      addItem: function(id, name, type, size) {
        var item = {id: id, name: name, type: type, size: size};
        this.selection.push(item);
        this.onChange();
      },

      removeItem: function(id) {
        this.selection = this.selection.filter(function(item) {
          return item.id !== id;
        });
        this.onChange();
      },

      onInit: function(cbInit) {
        var msg = {
          message: 'init',

```

```
        needAuthToken: true
    };
    _postMessage(msg);
    this.messageListener = this.onPostMessage.bind(this);
    window.addEventListener('message', this.messageListener, false);
    this.cbInit = cbInit;
},

onClose: function() {
    window.removeEventListener('message', this.messageListener, false);
},

onPostMessage: function(event) {
    if (event.data && event.data.message === "init") {
        if (this.cbInit) {
            this.cbInit(event.data);
        }
    }
},

onChange: function() {
    var msg = {
        message: 'change',
        selection: this.selection
    };
    _postMessage(msg);
},

onOk: function(selection) {
    selection = selection || this.selection;
    var msg = {
        message: 'ok',
        selection: this.selection
    };
    _postMessage(msg);
},

onCancel: function() {
    var msg = {
        message: 'cancel'
    };
    _postMessage(msg);
}
};
}

})();
```

This JS file needs to be packaged in a content connector.

This library follows the JS `postMessaging` paradigm to pass information. The following methods are of interest.

onInit

The custom picker needs to invoke this method during its initialization. This is where the custom picker can get connector configuration information, like client ID and access token. The attributes defined in the connector configuration (`/rest/api/v1/server`) are passed via this method.

```
OCE.CustomPicker.onInit(function(data) {
    {
        if (data != null) {
            self.clientId = data.ClientID;

            //If connector uses OAuth
            self.AccessToken = data.AccessToken;

            //If connector uses BASIC auth
            self.user = data.UserName;
            self.password = Base64.decode(data.UserPwd);
        }

        //Use client id and access token to fetch data from back end.
    }
})
```

onOk

If the custom picker exposes its own **OK** button, then it needs to invoke this method to pass back the event to Oracle Content Management.

```
function pickerCallback(data) {
    if (data[google.picker.Response.ACTION] ===
google.picker.Action.PICKED) {
        data[google.picker.Response.DOCUMENTS].forEach(function(doc) {
            OCE.CustomPicker.addItem(doc.id, doc.name, "file",
doc.sizeBytes);
        });
        OCE.CustomPicker.onOk();
    }
    else if (data[google.picker.Response.ACTION] ===
google.picker.Action.CANCEL) {
        OCE.CustomPicker.onCancel();
    }
}
```

onCancel

If the custom picker exposes its own **Cancel** button, then it needs to invoke this method to pass back the event to Oracle Content Management, like the **onOk** method does for the **OK** button in the preceding snippet.

addItem

This method needs to be invoked to pass back selections from the custom picker to Oracle Content Management so that it displays the list of selected items in the **Add to**

repository dialog. The IDs passed here are then used to fetch content with `/rest/api/v1/content` while adding an item to the asset repository.

```
addImage(selectedImages: Image[]) {
    selectedImages.forEach(o => {
        var name = "PexelsImages_" + o.uri;
        OCE.CustomPicker.addItem(o.uri, name, o.thumbnail, o.size);
    });
}
```

removeItem

This method needs to be invoked when an item is unselected from a custom picker so that the list maintained by Oracle Content Management is up to date. The final list is shown in the **Add to repository** dialog. The ID passed here is the same as the ID passed in the `addItem` call.

```
removeImage(image: Image) {
    OCE.CustomPicker.removeItem(image);
}
```

Package the custom picker in the connector. While registering the content connector, specify the following values:

1. **Custom Picker URL:** Such as `http://host:port/pexels-picker/web` (This is the custom picker packaged in the content connector.)
2. **Custom Picker uses its own OK/Cancel buttons:** Leave it unchecked. (This indicates that you want Oracle Content Management to embed your picker in their dialog.)

If the custom picker implements its own **OK** and **Cancel** buttons, then check this.

Authorization

A content connector can support one of the following authorization models.

- No Auth
- OAuth
- Basic

No Auth

This is used when the content connector does not require any authorization. In this case, the connector picker is launched directly without invoking any authorization screen.

ServerResource Implementation

```
serverInfo.authenticationType = AuthenticationType.NO_AUTH
```

OAuth

This is used when a content connector supports OAuth. In this case, OCM ensures that the OAuth flow is invoked to fetch the access token. For subsequent access, the already fetched token is used until it expires.

serverResource Implementation

```
serverInfo.authenticationType = AuthenticationType.OAUTH
```

The content connector also needs to define the required custom fields used in the OAuth flow.

ServerResource Implementation

```
{
    FieldInfo field = new FieldInfo();

    field.ID = UnsplashAdapter.FIELD_ID_REFRESH_TOKEN;
    field.label =
ResourceBundleUtil.getDefaultLocalizedString("cds.unsplash.adapter.fiel
d.refresh.token.label");
    field.labelLocalizations =

ResourceBundleUtil.getLocalizedData("cds.unsplash.adapter.field.refresh
token.label");
    field.datatype = FieldDatatype.STRING;
    field.userSettable = false;
    field.siteSettable = false;
    field.connectorSettable = true;
    field.authorizationURLParameter = false;

    serverInfo.fields.add(field);
}

{
    FieldInfo field = new FieldInfo();

    field.ID = UnsplashAdapter.FIELD_ID_ACCESS_TOKEN;
    field.label =
ResourceBundleUtil.getDefaultLocalizedString("cds.unsplash.adapter.fiel
d.access.token.label");
    field.labelLocalizations =

ResourceBundleUtil.getLocalizedData("cds.unsplash.adapter.field.accesst
oken.label");
    field.datatype = FieldDatatype.STRING;
    field.userSettable = false;
    field.siteSettable = false;
    field.connectorSettable = true;
    field.authorizationURLParameter = false;

    serverInfo.fields.add(field);
}
```



```

{
    FieldInfo field = new FieldInfo();

    field.ID = UnsplashAdapter.FIELD_ID_AUTHORIZATION_URL_PARAMETER_CODE;
    field.label =
ResourceBundleUtil.getDefaultLocalizedString("cds.unsplash.adapter.field.auth
url.code.label");
    field.labelLocalizations =

ResourceBundleUtil.getLocalizedData("cds.unsplash.adapter.field.authurl.code.
label");
    field.datatype = FieldDatatype.STRING;
    field.userSettable = false;
    field.siteSettable = false;
    field.connectorSettable = false;
    field.authorizationURLParameter = true;

    serverInfo.fields.add(field);
}

{
    FieldInfo field = new FieldInfo();

    field.ID = UnsplashAdapter.FIELD_ID_AUTHORIZATION_URL_PARAMETER_ERROR;
    field.label =

ResourceBundleUtil.getDefaultLocalizedString("cds.unsplash.adapter.field.auth
url.error.label");
    field.labelLocalizations =

ResourceBundleUtil.getLocalizedData("cds.unsplash.adapter.field.authurl.error
.label");
    field.datatype = FieldDatatype.STRING;
    field.userSettable = false;
    field.siteSettable = false;
    field.connectorSettable = false;
    field.authorizationURLParameter = true;

    serverInfo.fields.add(field);
}

```

And also implement the following interfaces documented in the preceding text:

1. /rest/api/v1/authorization/authorizationURLs
2. /rest/api/v1/authorization/completedAuthorizations

Basic

This is used when content connector requires login credentials to connect to the back end. Here OCM will prompt you to enter sign-in details before launching the picker for the first time. If the credentials are available, then they will be used.

**Note:**

Oracle no longer supports Basic authorization for external consumption.

ServerResource Implementation

```
serverInfo.authenticationType = AuthenticationType.BASIC
```

You also need to define all the login fields in the server implementation. These fields will display in the login screen.

```
// custom field for User Name
{
FieldInfo field = new FieldInfo();
field.ID = "UserName";
field.label =
ResourceBundleUtil.getDefaultLocalizedString("cds.unsplash.
adapter.field.username.label");
field.labelLocalizations = ResourceBundleUtil.getLocalizedData("
cds.unsplash.adapter.field.username.label");
field.description = ResourceBundleUtil.getDefaultLocalized-
String("cds.unsplash.adapter.field.username.desc");
field.descriptionLocalizations = ResourceBundleUtil.getLocalizedData("
cds.unsplash.adapter.field.username.desc");
field.datatype = FieldDatatype.STRING;
field.userSettable = true;
field.siteSettable = false;
field.connectorSettable = false;
field.authorizationURLParameter = false;
field.required = true;
serverInfo.fields.add(field);
}
// custom field for password
{
FieldInfo field = new FieldInfo();
field.ID = "UserPwd";
field.label =
ResourceBundleUtil.getDefaultLocalizedString("cds.unsplash.
adapter.field.password.label");
field.labelLocalizations = ResourceBundleUtil.getLocalizedData("
cds.unsplash.adapter.field.password.label");
field.description = ResourceBundleUtil.getDefaultLocalized-
String("cds.unsplash.adapter.field.password.desc");
field.descriptionLocalizations = ResourceBundleUtil.getLocalizedData("
cds.unsplash.adapter.field.password.desc");
field.datatype = FieldDatatype.PASSWORD;
field.userSettable = true;
field.siteSettable = false;
field.connectorSettable = false;
field.authorizationURLParameter = false;
field.required = true;
```

```
serverInfo.fields.add(field);  
}
```

Besides this, the content connector needs to implement the following interface, as described in the preceding text.

```
/rest/api/v1/authorization/basicAuthorization
```

Subsequently the user credentials (password base64 encoded) will be passed via headers for other calls, like filesystem and get content.

Content Connector Configuration and Registration

Once you have built your content connector, you need to register it in Oracle Content Management. You can register a content connector through the Oracle Content Management administration web interface.

The minimum properties you are required to add to a content connector are the following:

- Content connector name
- Content connector service URL
- User name and password, if the preceding URL access requires it

Once a content connector gets registered successfully and the content connector service URL is reachable, the Oracle Content Management server will call a server service on the content connector to get the custom properties and show those in the administration UI. An administrator or developer can add values for those properties (for example, `clientid` and `clientsecret` for OAuth flows or the user name and password for a Basic authorization central account). Clicking **Save** saves all these properties with the connector framework. After you enable it, the content connector becomes available in the **Add** menu item of the Oracle Content Management UI **Assets** page.

See *Create and Configure a Custom Content Connector* in *Administering Oracle Content Management* and [Register the Content Connector](#), which describes how to register and configure a Pexel content connector.

Content Connector Execution Flow

From the Oracle Content Management UI **Assets** page, the user can click **Add** and choose a content connector from the menu.

Based on the authorization type defined, the OAuth flow is triggered or the user is prompted to add a user name and password for the Basic Auth flow (if a central account has not been configured in the administration UI). For No Auth, the content connector picker is launched directly without invoking any authorization screen.

After successful authorization, the configured file picker (either common or custom) is launched. It lists the files available to the user from the remote store.

The user can select one or more files. The selected files list is passed to the connector framework. The connector framework then makes REST API calls to the remote content connector to fetch the content of the assets being added and copies them into the asset repository. This happens as a background process; however, the status of the add is

displayed in the UI where success and failure can be tracked for each asset added. Refer to the diagnostic logs for any failures.

Pexels Content Connector Sample Implementation

You can use this sample implementation of a Pexels content connector to help build your own custom content connectors. The Pexels content connector lets content creators bring rich, high-quality images from the Pexels library into an Oracle Content Management asset repository.

Pexels is a stock photography site where designers, bloggers, and others find photos to use for free. The Pexels content connector is built using the Connector Framework SDK provided by Oracle Content Management. Two versions of this content connector are available:

1. A content connector that uses a custom picker built using Angular JS. This version is built by default.
2. A content connector that uses the common picker provided by Oracle Content Management.

The following sections describe how to develop a Pexels content connector.

- [Install the Content Connector](#)
- [Register the Content Connector](#)
- [Test the Content Connector](#)
- [Understand the Content Connector Source Code](#)
- [Custom Picker UI](#)
- [Pexels REST APIs](#)
- [Change and Test the Content Connector Code](#)

Install the Content Connector

To install the Pexels content connector, you need to meet the installation prerequisites and then build the content connector WAR file.

The following sections describe the prerequisites and how to build the WAR file.

Check Prerequisites for Installation

Your system needs to be set up with node, npm, JDK, and Apache Maven before you install the Pexels content connector.

1. Ensure you have node and npm installed on your machine and set in PATH. Configure proxy if required.
See [Configure Proxy Service Settings](#).
2. Ensure you have latest JDK installed and set in PATH.
3. Ensure you have Apache Maven installed and set in PATH, with proxy configured if required.

Build the Content Connector WAR File

Download the content connector source bundle, unzip the bundle into a location on your machine, and follow the instructions in the `Readme` file to generate the WAR file.

You can download the Oracle Content Management content connector sample and SDK from here:

```
https://<your-cec-service>/_sitesclouddelivery/renderer/app/sdk/connector/cec-connector-sample.zip
```

```
https://<your-cec-service>/_sitesclouddelivery/renderer/app/sdk/connector/cec-connector-sdk.zip
```

Briefly, the instructions are as follows:

1. In a command-line interface, go to `pexelsPickerWeb\src\main\webapp`.
2. Run `pexels_setup.bat` or `pexels_setup.sh` (based on your environment).
3. If the run is successful, `pexels-picker.war` file will be available in the `pexelsPickerWeb\target` folder.

The WAR file generated is the one that uses the custom picker UI, which was built using Angular JS.

Deploy this WAR file to your server. Run the following URLs to test that it works well:

1. GET `http://<host:port>/pexels-picker/rest/api`
2. GET `http://<host:port>/pexels-picker/rest/api/v1/server`

For expected output, see [Understand the Content Connector Source Code](#).

Register the Content Connector

To use a Pexels content connector, you first need to access to the Pexels API. For this you need to register with Pexels and request access.

Follow the instructions in the Pexels API documentation to register the content connector and request access. After you make the request, you will be provided with an API key that you need to capture. The Pexels API is rate-limited to 200 requests per hour and 20,000 requests per month. If you need higher limits, contact Pexels.

1. Sign in to your Oracle Content Management instance as a developer or administrator.
2. Go to **Administration > Integration > Content Connectors** and click **Create**.
3. Enter the details to register your Pexels content connector.
 - a. **Name:** Pexels
 - b. **Description:** Connector to bring Pexels images into Oracle Content Management asset repository
 - c. **Connector Service URL:** `http://<host:port>/pexels-picker/rest/api` (a URL tested previously)
4. Click **Verify Settings** and then **Save**.

5. On the **Content Connectors** tab, click **Configure** to go back to the content connector configuration and provide the following values.
 - a. **Custom Picker URL:** `http://<host:port>/pexels-picker/web/index.html`
This is the custom picker packaged in a content connector. This setting is not applicable when you use the common UI.
 - b. **Hide OK/Cancel:**
Leave this setting unchecked. It indicates whether or not you want Oracle Content Management to embed your picker in their dialog, which is not applicable when you use the common UI.
6. Also provide the **Client ID** in the **Custom Fields** section.
7. Check **Enable for End Users**.

The default timeout values for the content connector are set in the following two properties:

- `ConnectorConnectionTimeout=20000`
- `ConnectorReadTimeout=30000`

If you want to change the values of these properties, you can add the properties to your `config.cfg` and then modify either or both values.

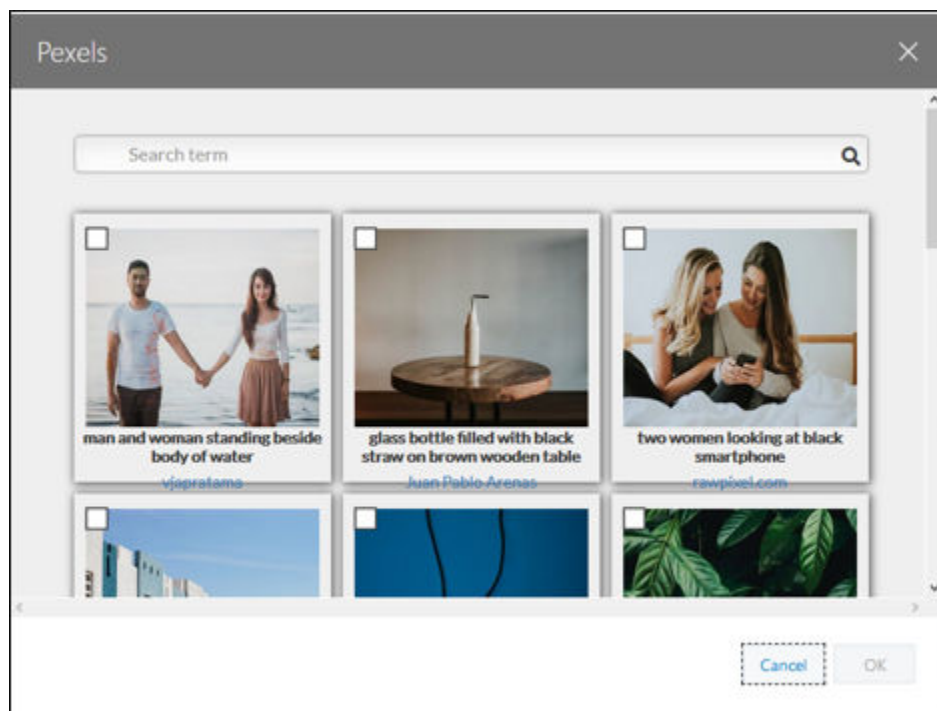
Test the Content Connector

Verify the functionality of your Pexels content connector.

To test the content connector, go to the **Assets** tab and click **Add**. You should see **Import from Pexels** in the drop-down menu. Ensure that you have created an asset repository and selected it before you click **Add**. This will launch the Pexels picker showing curated images up front.

This picker is either the custom UI packaged in the content connector or the common picker, depending on which you installed.

You can search for required images, select the ones you are interested in, and click **OK**. This will open up the **Add to Repository** dialog, which lists the selected images you can add. After you add them, the images should appear in the asset repository.



Understand the Content Connector Source Code

The source code of the Pexels content connector contains REST APIs and a custom picker.

The Pexels content connector contains a set of REST APIs implemented using the Oracle Content Management content connector interfaces. This is implemented as per the JAX-RS specification.

The Connector SDK and sample implementation are in the following files:

- `cec-connector-sdk.zip`
This file contains the SDK interface JAR, Javadoc, and `connector.yaml` file, which describes the REST APIs and their payloads.
- `oracle-ocm-custompicker.js`
This is the JS file that the content connector custom UI must package to interact with Oracle Content Management.
- `pexelsPickerWeb.zip`
This file contains the Pexels content connector reference implementation.

Custom Picker UI

The Pexels picker packaged in the content connector is built using Angular JS. It provides the Custom Picker UI.

The Pexels picker files are in `pexelsPickerWeb\src\main\webapp\src`.

Pexels REST APIs

The Pexels content connector contains a set of REST APIs.

Change and Test the Content Connector Code

If you make changes to the content connector code, you can run commands from `pexelsPickerWeb\src\main\webapp` to test the changes and rebuild the WAR file.

1. If you modify `package.json` to include additional libraries, run the following command from `pexelsPickerWeb\src\main\webapp`:

```
npm install
```

2. If you make UI code changes, run the following command from `pexelsPickerWeb\src\main\webapp`:

```
ng build
```

3. Finally, run `mvn` from `pexelsPickerWeb` to rebuild the WAR file:

```
mvn clean install
```

Download the CEC Content Connector Sample and SDK

A sample content connector and SDK are available for download

You can download the following ZIP files, which contain the CEC content connector sample and SDK:

- https://<your-cec-service>/_sitesclouddelivery/renderer/app/sdk/connector/cec-connector-sample.zip
- https://<your-cec-service>/_sitesclouddelivery/renderer/app/sdk/connector/cec-connector-sdk.zip

7

Develop Custom Field Editors

Oracle Content Management provides a large selection of field editors out of the box for data fields that you can use to enable contributors to assign values to the field. Oracle Content Management also allows you to extend beyond out of the box editors with the help of the field editor SDK.

With the Oracle Content Management field editor SDK, you can develop and use your own custom field editors for any field. The main purpose of the SDK is to facilitate communication between the custom editor component and the content item form. To develop a custom field editor, you first need to create the custom field editor. Once created, you must promote it so that it can be used when creating content types. Once promoted, the field editor is available for use when a data type named when creating the editor is used to define a content type. Lastly, you can open and edit the files associated with the custom field editor to fit your purpose.

You can make custom field editors and forms available to everyone:

1. Update existing content forms and field editors to public during server upgrade in the background.
2. Add the service `SCS_SET_CONTENT_FORMS_FIELD_EDITORS_PUBLIC` to set all form and editor folders to public.
3. In `SCS_CREATE_COMPONENT`, mark `content form/field editor` as public on creation.
4. Mark `content form/field editor` as public on import.

The following sections provide more information about custom field editors.

- [Create a Custom Field Editor](#)
- [Configure Content Type to Use Custom Field Editor](#)
- [Edit a Custom Field Editor](#)
- [Sample Content Field Editors](#)
- [Content Field Editor SDK Reference](#)

Create a Custom Field Editor

To create a custom field editor:

1. Log in to your instance of Oracle Content Management using the web interface and click **Developer** in the side navigation.
2. Click **View All Components**.
3. Click **Create** and select **Create Content Field Editor**.
4. Enter a name and optional description for the field editor.
5. If your field editor can handle multiple values, such as multi-valued select boxes or tag cloud editors, enable **Handles multiple values**.

6. Select the data fields to be used with the field editor. The following data types are supported:
 - Text
 - Large Text
 - Date
 - Number
 - Decimal
 - Boolean
 - Embedded Content
7. Click **Create**. A new plain text field editor is created and listed on the Components page.
8. To make the field editor available for use, select it and click **Promote**.
9. Confirm you want to proceed and click **OK**.

The custom field editor is now available for use with supported data fields when defining content types.

Configure Content Type to Use Custom Field Editor

Once a custom field editor has been created and promoted, it is available for use when supported data types are used to define a content type.

1. In Oracle Content Management, click **Content** in the side navigation.
2. Select Asset Types from the page menu and click **Create**.
3. Enter a name and optional description and click **OK**.
4. Select a data field that the custom editor can use and add it to the definition.
5. Enter the desired information and click **Next**.
6. Select the custom field editor from the **Appearance of data field** and click **OK**.

Any content items created using this content type will make use of the custom field editor.

Edit a Custom Field Editor

Once you've created a custom field editor, the files needed to adapt it to your needs are available when you select and open the component.

1. Click **Developer** in the side navigation.
2. Click **View All Components**.
3. Select the custom field editor component and click **Open**.

By default, a custom field editor comes with the following content:

- [appinfo.json](#) - contains the metadata of the editor
- assets folder that includes two files which implement the custom field editor:
 - [edit.html](#)

- [view.html](#)
- `folder_icon.png` - this icon the editor uses when displayed on the components page.

appinfo.json for Custom Field Editors

This file is a required and includes the configuration metadata needed for the editor. Some properties are necessary for the editor to function, some are optional. The following is a sample `appinfo.json`:

```
{
  "type": "fieldeditor",
  "name": "Sample Text field",
  "single": true,
  "multiple": true,
  "handlesMultiple": false,
  "supportedDatatypes": ["text"],
  "autoresize": true,
  "validation": true,
  "useDefaultFormView": false
}
```

Table 7-1 `appinfo.json` Configuration Properties

Property	Value	Required	Type
<code>type</code>	The component type. For custom field editors, the type is <code>fieldeditor</code> and is the same for all editors. It should not be changed.	required	string
<code>name</code>	The name displayed in the Content Type field settings editor list. By default the name entered when the custom field editor is created.	required	string
<code>single</code>	Specifies if this editor will be listed in the editor list when the field is a single valued field.	Either single or multiple is required and set to true	Boolean
<code>multiple</code>	Specifies if this editor will be listed in the editor list when the field is multivalued field.	Either single or multiple is required and set to true	Boolean
<code>handlesMultiple</code>	Specifies if this editor can handle more than one value.	required	Boolean
<code>supportedDataTypes</code>	List of data types for which this editor can be used.	required	Array of data types
<code>autoresize</code>	Specifies whether or not to send resize notification to the content item form when this editor is resized. Custom editors are rendered inside an <code>iframe</code> in the content item form. When this property is set to true, the field editor SDK will notify the form to adjust the <code>iframe</code> size when the editor's size changes.	optional	Boolean

Table 7-1 (Cont.) appinfo.json Configuration Properties

Property	Value	Required	Type
validation	Specifies whether or not the value entered in this editor needs to be validated in the content item editor form. When set to true, the editor needs to register and provide a call back function to be called by the content item form for validation.	optional	Boolean
useDefaultFormView	Specifies whether or not to use the system default form view when viewing the content item that has this editor. If this property is not present or is set to false, the view.html implementation will be used to render the view mode of the field.	optional	Boolean

In addition to the above properties, you can also define settings that can be used to render the editor configuration view when this editor is selected while defining a content type. For example, if you are writing an editor to handle a phone number in your field and want to set up a few formats to pick during field definition time:

```
{
  "type": "fieldeditor",
  "name": "Phone Number",
  "single": true,
  "multiple": true,
  "handlesMultiple": false,
  "supportedDatatypes": ["text"],
  "autoresize": true,
  "useDefaultFormView": false,
  "settings": [
    {
      "id": "mask",
      "type": "list",
      "name": "Phone number format",
      "options": [
        {
          "label": "+1(999)-999-9999",
          "value": "+1(999)-999-9999"
        },
        {
          "label": "+44(999)-9999-9999",
          "value": "+44(999)-9999-9999"
        },
        {
          "label": "+49 999 99999999",
          "value": "+49 999 99999999"
        }
      ]
    }
  ]
}
```

Note that settings in an array can have more than one type. Currently two types of settings are supported:

- text
- list

Text

The text type settings have three properties:

- **id**: the field ID
- **name**: the field name
- **type**: the field type

For example:

```
{
  "id": "min",
  "name": "Minimum",
  "type": "text"
}
```

would render an editor configuration view with a text field named **Minimum** in the settings dialog with an ID of `min`

If you want to use a default value for the field as part of the settings, you can modify the settings:

```
{
  "id": "oracle.cloud.content.defaultValue",
  "type": "text"
}
```

where now the default value field is displayed.

List

For a list, in addition to `id`, `type`, and `name`, a list of options for display in the list also needs to be provided. For example, if you are writing an editor to handle a phone number in your field and want to set up a few formats to pick during field definition:

```
{
  "type": "fieldeditor",
  "name": "Phone Number",
  "single": true,
  "multiple": true,
  "handlesMultiple": false,
  "supportedDatatypes": ["text"],
  "autoresize": true,
  "useDefaultFormView": false,
  "settings": [
    {
      "id": "mask",
      "type": "list",

```

```
"name": "Phone number format",
"options": [
  {
    "label": "+1(999)-999-9999",
    "value": "+1(999)-999-9999"
  },
  {
    "label": "+44(999)-9999-9999",
    "value": "+44(999)-9999-9999"
  },
  {
    "label": "+49 999 99999999",
    "value": "+49 999 99999999"
  }
]
}
]
```

edit.html for Custom Field Editor

The edit.html file is where the custom field editor implementation goes. It is a mandatory file. In general it deals with html markup, styles and JavaScript code needed for the editor. This file loads the field-editor-sdk-1.0.js JavaScript library in order to communicate with the content item form. The edit.html file loads in an iframe inside the content item form for the field for which the custom field editor is configured. When you create a custom field editor, the default custom editor is a simple text field that demonstrates how the field editor can render within the form and get and set field values. If the custom field editor is configured for a multi-valued field, then the editor is implemented within the iframe for each entry.

To adapt a custom field editor you need to:

1. Edit the HTML markup
2. Import the SDK library
3. Get and set field values
4. Validate field values
5. Enable and Disable field editor

Edit the HTML Markup

Open edit.html in a text editor and make the required changes. For example:

```
<html>
  <head>
    <style>
      body {
        margin: 0;
        padding: 0;
        font: normal 100% "Helvetica Neue", 'Segoe UI', sans-serif-
regular, Helvetica, Arial, sans-serif;
      }
      input {
```

```

        width: 100%;
        height: 32px;
        font-size: 14px;
    }
</style>
</head>
<body>
    <!-- In this case the editor is a text field -->
    <input id="textInput" type="text">
</body>
    .....
</html>

```

Import the SDK Library

Import the `field-editor-sdk-1.0.js` library either using the `<script>` tag or `requireJS`. This SDK has a global object called `editorSDK`.

```

<!-- load the field editor SDK -->
<script src="/documents/static/gemini/api/field-editor-sdk-1.0.js"></script>

```

Get and Set Field Values

Initialize the `editorSDK` and pass a callback function and optional container DOM element. This callback function is called by the `editorSDK`'s `initSDK()` method with an instance of the SDK. With the SDK instance, the field being edited can be accessed. With the field instance in place, the value of the field being edited can be obtained and a new value can be set.

Below is the sample callback function `initEditor` that gets the field object corresponding to this custom editor from the SDK instance, then gets the value of the field and sets in the text field editor. It then listens to field value change, and updates field's value.

```

<script>
    /* globals editorSDK */
    (function() {

        function initEditor(sdk) {
            // retrieve the field object rendered by this custom editor
            var field = sdk.getField();

            // retrieve the current field value
            var value = field.getValue();

            var inputField = document.getElementById('textInput');

            // set the current field value in this editor
            inputField.value = value;

            // when the editor value changes, set the changed value
            // to the field
            inputField.addEventListener('change', function(e) {
                field.setValue(inputField.value);
            });
        }
    });

```

```
        // if the field is updated externally, keep the editor in
sync
        field.on('update', function(value) {
            inputField.value = value ? value : '';
        });
    }

    // this is the entry point to initialize the editor sdk.
    editorSDK.initSDK(initEditor);

    }) ();
</script>
```

Validate Field Values

In addition to getting and setting field values, if the custom field editor needs to validate the values entered, then it can use the `sdk.setValidation()` function and pass a callback function that handles the editor validation. In the example below, `setValidation` passes the callback function `validateValue`. This works when the validation property is set to `true` in the `appinfo.json` file.

```
    // sample validation function that
    // validates whether entered length of the string is more than
10
    function validateValue(value) {
        var isValid = true;

        if (value && value.length > 100) {
            isValid = false;

            return {
                isValid: false,
                title: getTranslatedString('validation.title',
currentLocale),
                message: getTranslatedString('validation.message',
currentLocale)
            };
        }
        return isValid;
    }

    function initEditor(sdk) {
        // retrieve the field object rendered by this custom editor
        var field = sdk.getField();
        ....
        ....

        // register a custom validation function for this editor
        sdk.setValidation(validateValue);
        ....
        ....
    }
```



```
// this is the entry point to initialize the editor sdk.  
editorSDK.initSDK(initEditor);
```

Enable and Disable Field Editor

Content item form field editors get enabled and disabled. For example, each field editor goes into the disabled mode when annotation starts, and when the user comes out of annotation mode the form field editors get enabled again. Custom field editors also need to follow this rule, so the custom field editor implementation should handle the enabling and disabling of the editors when the form broadcasts to do so. This is handled by calling the `registerDisable` method of the SDK and passing a callback function that handles enabling the editor.

```
//  
// disable/enable call back that gets called if the content form  
// needs to enable/disable this editor  
function renderDisabled(value) {  
    if (value) {  
        document.getElementById('textInput').disabled = true;  
    } else {  
        document.getElementById('textInput').disabled = false;  
    }  
}  
  
function initEditor(sdk) {  
    // retrieve the field object rendered by this custom editor  
    var field = sdk.getField();  
    ....  
    ....  
  
    //register render disabled function for this editor  
    sdk.registerDisable(renderDisabled);  
    ....  
    ....  
}  
  
// this is the entry point to initialize the editor sdk.  
editorSDK.initSDK(initEditor);
```

view.html for Custom Field Editor

The `view.html` contains html markup, styles and JavaScript code for the custom viewer implementation. This file is executed for the field using the custom field editor when the user switches the content form to view mode. The `view.html` file gets called only when `useDefaultFormView` is not present or set to `false` in the `appinfo.json` file, otherwise the system default viewer is used for the custom field editor. Unlike the `edit.html` file, the `view.html` file is invoked only once whether or not the field is single valued or multi-valued.

Below is a sample `view.html` file that displays the field value within an inner HTML of a div element.

```
<!DOCTYPE html>  
<html>  
  <head>
```

```
<meta charset="utf-8">
<style type="text/css">
  body {
    margin: 0;
    padding: 0;
    font: normal 100% "Helvetica Neue", 'Segoe UI', sans-serif-
regular, Helvetica, Arial, sans-serif;
    color: #666;
  }
  div {
    font-size: .875rem;
  }
  span {
    font-style: italic;
  }
</style>
</head>
<body>
  <div id="inputValue"></div>
</body>

<!-- load the attribute editor SDK -->
<script src="/documents/static/gemini/api/field-editor-sdk-1.0.js"></
script>
<script>
/* globals editorSDK */
(function() {
var textElement = document.getElementById('inputValue');

function initEditor(sdk) {
  // retrieve the field object rendered by this custom editor
  var field = sdk.getField();

  // retrieve the current field value
  var value = field.getValue();

  if (value) {
    textElement.innerText = value;
  } else {
    textElement.innerHTML = '<span>No value specified</span>';
  }
}
editorSDK.initSDK(initEditor);
})();
</script>
```

Sample Content Field Editors

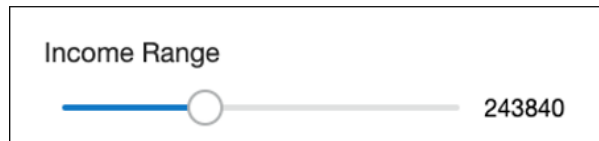
Oracle Content Management provides 2 sample content field editors available from Content Toolkit.

- [Slider](#)
- [Location Selector](#)

These sample field editors can be exported from Content Toolkit and imported to your instance of Oracle Content Management like any other component.

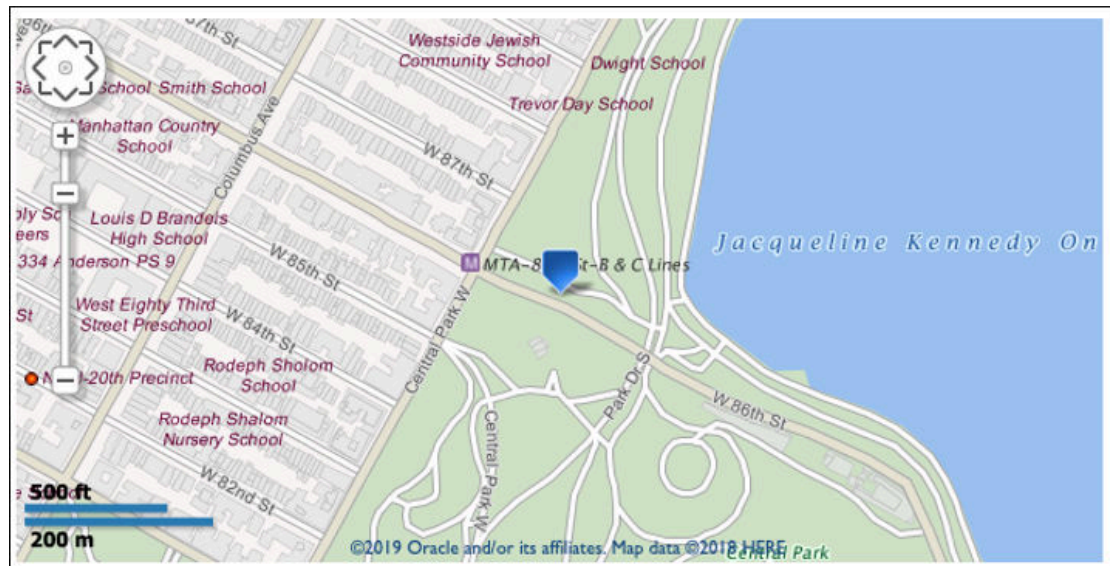
Slider

The slider sample is built using the Oracle Jet Slider widget. It can be used to pick a number in a give range. This is useful for the 'NUMBER' data type. Minimum and maximum values of the range can be configured. This sample also demonstrates how the *field-editor-sdk* library can be used by using requireJS. The slider sample is compatible for single valued and multi-valued fields.



Location Selector

The location selector sample content field editor is built using Oracle's [e-location JavaScript library](#). It can be used to pick a location on a map. It is used with the 'TEXT' data type. Default longitude, latitude and zoom level for the map can be configured for this editor. This location selector sample sets the selected location's longitude and latitude as comma-separated values in the custom field. It also displays the selected location on the map with a pointer.



Content Field Editor SDK Reference

The Oracle Content Management content field editor SDK API enables communication between a custom field editor and a content item defined with a content type that uses the custom field.

The content field editor SDK has a global object called `editorSDK` that initializes the field editor. The entry point is to call the `editorSDK.initSDK` and passes a callback function like `editorSDK.initSDK(initEditor)`.

There are two objects:

- [SDK Object](#)
- [Field Object](#)

Content Field Editor SDK Object

The `initEditor` is the callback function that gets an instance of the SDK object that provides the following methods:

getField()	Gets a field object rendered by the custom editor.
getFields()	Gets a list of all fields in the content item in which the custom editor is being used. Returns an array of field objects.
getLocale()	Gets the current language of the user interface.
getSetting(setting)	Gets the editor setting value for the given setting.
getSettings()	Gets the editor settings.
registerDisable(callback)	Registers a callback function for disabling and enabling the custom field editor.
setValidation(callback)	Registers a validation callback function for validating the custom field editor.
resize(size)	Resizes the iframe where the custom editor resides to the given size object containing width and height in pixels.
openContentPicker()	Opens the content picker dialog.
getDirection()	Gets the current direction of the user interface.

getField()

Parameters:

None

Usage:

```
// retrieve the field object rendered by this custom editor
var field = sdk.getField();
```

Returns:

Field object

getFields()

Parameters:

None

Usage:

```
// retrieve all fields
var fields = sdk.getFields();
```

Returns:

An array of Field objects

getLocale()

Parameters:

None

Usage:

```
// get the locale the ui is currently in
currentLocale = sdk.getLocale();
```

Returns:

The language code of the user that is logged in

getSetting(setting)

Name	Type	Description
setting	string	Name of the setting

Usage:

```
// get the initial latitude from settings
var latitude = sdk.getSetting('init-lat');
```

Returns:

Value of the setting. For example if the settings property for custom editor in appinfo.json is:

```
"settings":[
  {
    "id": "foo",
    "type": "text",
    "name": "Foo",
  },
  ...
  ...
]
```

Then this 'foo' setting will appear in the field definition for content administrator to configure. Suppose if the content admin configured this setting to have a value 'bar', then `sdk.getSetting('foo')` will return 'bar'.

getSettings()

Parameters:

None

Usage:

```
//get editor settings
var settings = sdk.getSettings();
```

Returns:

Editor settings configured by content admin for the custom field editor. For example, in the case of the sample location selection editor, if the setting is:

```
"settings": [{
  "id": "init-long",
  "name": "Initial longitude",
  "type": "text"
}, {
  "id": "init-lat",
  "name": "Initial latitude",
  "type": "text"
}, {
  "id": "zoom-level",
  "name": "Zoom level",
  "type": "text"
}]
```

and if the content admin entered an initial longitude of -73.96 and an initial latitude of 40.78, with a zoom level of 18, then `sdk.getSettings()` will return:

```
{zoom-level: "18", init-long: "-73.96", init-lat: "40.78"}
```

registerDisable(callback)

Name	Type	Description
callback	function	Function that handles the enabling and disabling of the editor

Usage:

```
//register render disabled function for this editor
sdk.registerDisable(function renderDisabled(value) {
  if(value) {
    document.getElementById("phoneNumber").disabled = true;
  }else {
    document.getElementById('phoneNumber').disabled = false;
  }
});
```

```
    }
  });
```

setValidation(callback)

Name	Type	Description
callback	function	Function that handles field value validation

Usage:

```
// register a custom validation function for this editor
sdk.setValidation(function validateValue(value) {
  // sample validation function that
  // validates whether entered number of characters is more than 100
  var isValid = true;

  if (value && value.length > 100) {
    isValid = false;

    return {
      isValid: false,
      title: 'Invalid value',
      message: 'You have entered more than 100 characters'
    };
  }
  return isValid;
});
```

resize(size)

Name	Type	Description
size	Object with height and width	Object with height and width in pixels. For example: <pre>{ height: '20px', width: '60px' }</pre> <p>If no size is given then the size of the editor will be used.</p>

Usage:

```
sdk.resize({width: '100px', height: '20px'});
```

openContentPicker()

Opens the content picker dialog.

Usage:

```

    sdk.openContentPicker().then(function(data) {
        // use data returned from content picker
        // sample data {id: 'assetId', type: 'assetType', name:
'assetName'}
        console.log(data)

        }).catch(function(error){
            // handle error
            console.log("Error:", error);
        });

```

Returns:

A Promise object.

getDirection()

Parameters:

None

Usage:

```

// when the editor wants to handle and support
// directionality
let dir = sdk.getDirection();
if(dir){
document.querySelector("html").setAttribute("dir", dir);
}

```

Returns:

The current direction of the user interface.

Custom Content Field Object

The Field object is the representation of the content field that is returned by `getField()` and `getFields()` methods. The object has the following methods:

<code>getName()</code>	Gets the name of the field.
<code>getDefaultValue()</code>	Gets the default value of the field.
<code>getDataType()</code>	Gets the data type of the field.
<code>setValue(value)</code>	Sets the value of the field.
<code>getValue()</code>	Gets the value of the field.
<code>on(event, callback)</code>	Attaches an event handler function to the field for the event. Currently <code>update</code> is the only event supported.

getName()

Parameters:

None

Usage:

```
// retrieve the field object rendered by this custom editor and get its  
name  
var field = sdk.getField();  
var fieldName = field.getName();
```

Returns:

Name of field

getDefaultValue()

Parameters:

None

Usage:

```
// retrieve the field object rendered by this custom editor and get its  
default value  
var field = sdk.getField();  
var fieldDefaultValue = field.geDefaultValue();
```

Returns:

Default value if one is set for the field.

getDataType()

Parameters:

None

Usage:

```
// retrieve the field's data type  
var field = sdk.getField();  
var dataType = field.getDataType ();
```

Returns:

Data type of the field, for example, text or number

setValue(value)

Name	Type	Description
value	string, number, or object, depending on the data type of the field	Varies depending on the data type of the field

Usage:

```
// when the editor value changes, set the changed value
// to the field
inputField.addEventListener('change', function(e) {
  field.setValue(inputField.value);
});
```

getValue()

Parameters:

None

Usage:

```
// retrieve the field object rendered by this custom editor and
get its value
var field = sdk.getField();

var fieldValue = field.getValue();
```

Returns:

Value of the field

on(event, callback)

Name	Type	Description
event	Event string	Event string. Currently only the event update is supported.
callback	Function	Event handler function to be attached to the event

Usage:

```
// if the field is updated externally, keep the editor in sync
field.on('update', function(value) {
  inputField.value = value ? value : '';
});
```

8

Develop Custom Content Forms

Oracle Content Management allows you to extend beyond out of the box forms with the help of the content form SDK.

With the Oracle Content Management content form SDK, you can develop and use your own custom form. The main purpose of the SDK is to facilitate communication between the custom form component and the content item form. To develop a custom form, you first need to create the custom form then promote it so that it can be used when creating content types. Once promoted, the form is available for use. Lastly, you can open and edit the files associated with the custom form to fit your purpose.

- [Create a Custom Content Form](#)
- [Configure a Content Type to Use a Custom Content Form](#)
- [Edit a Custom Content Form](#)
- [Content Form SDK Reference](#)
- [Sample Custom Form](#)

Create a Custom Content Form

To create a custom content form:

1. Log in to your instance of Oracle Content Management using the web interface, and click **Developer** in the side navigation.
2. Click **View All Components**.
3. Click **Create**, and select **Create Content Form**.
4. Enter a name and optional description for the form.
5. Pick a size of the drawer for the content form.
6. Click **Create**. A new starter form component is created and listed on the Components page.
7. To make the form available for use, select it and click **Promote**.
8. Confirm you want to proceed, and click **OK**.

The custom form is now available for use for the selected content type(s).

Configure a Content Type to Use a Custom Content Form

Once a content form has been created and promoted, it is available for use when defining a content type.

1. In Oracle Content Management, click **Content** in the side navigation.
2. Select **Asset Types** from the page menu, click one of the content types, and click **Edit**.

- To edit an existing type, select the content type to use the content form and click **Edit**.
 - To create a new content type, enter a name and optional description and click **OK**. Then define the content type.
3. On the **Content Layout** tab, select the content form from the **Content Form** field.
 4. Make any additional changes to the content type and when finished, click **Save**.

Any content items created or edited of this content type will make use of the content form.

Edit a Custom Content Form

Once you've created a custom content form, the files needed to implement the form are available when you select and open the component.

1. Click **Developer** in the side navigation.
2. Click **View All Components**.
3. Select the custom content form component and click **Open**.

By default, a custom content form comes with the following content:

- [appinfo.json](#) contains the metadata of the content form.
- `assets` folder includes files that implement the custom content form—`edit.html`.
- `folder_icon.png` is the icon that the content form uses when displayed on the components page.

appinfo.json

This file is a required file and includes the configuration metadata needed for the content form.

```
{
  "type": "contentform",
  "name": "Sample form",
}
```

Property	Value	Required	Type
type	The component type. For custom content forms, the type is <code>contentform</code> and is the same for all forms. It should not be changed.	required	string
name	Name of the custom content form	required	string

Property	Value	Required	Type
drawerSize	<p>Size of the drawer in which the custom form will appear. It can be one of the following Oracle Content Management UI's predefined drawer sizes:</p> <ul style="list-style-type: none">Default - Responsive at 90% of windowFull - Responsive at 100% of windowHalf - Responsive at 50% of windowSmall - Fixed width of 840px, expands/collapses when sidebar opens/closes.Medium - Fixed width of 1140px, expands/collapses when sidebar opens/closes.Large - Fixed width of 1296px, expands/collapses when sidebar opens/closes. <p>When the <code>drawerSize</code> property is not present or when no value is specified, 'Default' will be used.</p>	optional Default value is 'Default'	string
create	<p>The optional create property can be used to control the behavior of the sidebars while creating content items. The only valid property of the create object is a sidebar object. The valid properties for sidebar are the same as the <code>contentItemCreate</code> sidebar object from the Embed UI API V2 Reference. A shortcut to disable all the sidebars during create is to provide an empty sidebar object.</p> <pre>create: { sidebar: {} }</pre> <p>As all values are initialized, only the sidebar properties that are changed from the default value of the property need to be supplied in the sidebar object.</p>	optional	object

Property	Value	Required	Type
edit	<p>The optional edit property can be used to control the behavior of the sidebars while editing content items. The only valid property of the edit object is a sidebar object. The valid properties for sidebar are the same as the <code>contentItemEditor</code> sidebar object from the Embed UI API V2 Reference.</p> <p>A shortcut to disable all the sidebars during edit is to provide an empty sidebar object.</p> <pre>edit: { sidebar: {} }</pre> <p>As all values are initialized, only the sidebar properties that are changed from the default value of the property need to be supplied in the sidebar object.</p>	optional	object
usesCustomEditors	Whether the form uses custom field editors. Default is false	optional	boolean

edit.html for Custom Content Forms

The `edit.html` file is where the custom content form implementation begins. It is a mandatory file. This file is called to load the custom content form within the iframe when creating or editing a content item. In general it deals with HTML markup, styles, and code implementation needed for rendering the content item fields and metadata. This file loads the `content-form-sdk-1.0.js` JavaScript library to communicate with the content item page in the user interface. This file also handles notifying field value changes to the user interface and listening to the updates from the interface. The default implementation of this file in starter form renders system metadata field editors of the item. This can be edited to include type-specific field editors or to alter system field editors as needed.

Import the SDK Library

Oracle provides a JavaScript API for the custom form residing in an iframe element to communicate with the content item edit page in the user interface. So this library should be included in `edit.html`. Load the `content-form-sdk-1.0.js` library using either the `<script>` tag or `requireJS`. This SDK has a global object called `contentFormSDK`.

Using the `<script>` Tag

```
<!-- load the content form SDK -->
<script src="/documents/static/gemini/api/content-form-sdk-1.0.js"></script>
```

Using requireJS

```
<!-- using requireJS -->
<script>
  requirejs.config({
    paths: {
      contentFormSDK: '/documents/static/gemini/api/content-form-
sdk-1.0'
    }
  });
  require(['contentFormSDK'],
    function (contentFormSDK) {
  });
</script>
```

Initialize the Form

Invoke `contentFormSDK.init()` and pass a callback function to initialize the form and start communicating with the interface. This callback function will be called with an instance of the SDK when the user interface is ready to render the form. With the SDK instance, the item being created or edited can be accessed, and the custom form code can communicate with the interface.

Below is the sample callback function `initForm`.

```
<script>
  /* globals contentFormSDK */
  (function() {

    function initForm(sdk) {

      // type
      var type = sdk.getType();

      // item being rendered in the form
      var item = sdk.getItem();

      // current locale of the UI
      var locale = sdk.getLocale();

      // fields of the item
      var fields = item.getFields();

      // item properties
      var itemProps = item.get();
    }

    // this is the entry point to initialize the form sdk.
    contentFormSDK.init(initForm);

  }) ();
</script>
```

Get and Set Item Name and Description

The item name and description can be obtained from the `item.get()` method . When they are set in their respective editors, the change event listeners of the editors can set their updated value in the item by calling `item.setName()` and `item.setDescription()` as shown in the following sample.

```
// item properties
var itemProps = item.get();

// name field
var nameField = itemProps.name ? itemProps.name : '';
nameField.addEventListener('change', function(e) {
    item.setName(nameField.value);
});

//description field
var descField = document.getElementById('description-input');
descField.value = itemProps.description ?
itemProps.description : '';
descField.addEventListener('change', function(e) {
    item.setDescription(descField.value);
});
```

Get and Set Field Values

Custom fields can be obtained from the `item.getFields()` method. It returns an array of field objects. These field objects help access field values and modify them.

```
// fields of the item
var fields = item.getFields();

fields.forEach(function(field) {
    var fieldDefn = field.getDefinition(),
        fieldName = fieldDefn.name,
        dataType = fieldDefn.datatype;

    // handle updating a field of datatype 'text'
    if(dataType === sdk.dataTypes.TEXT) {
        var titleInput = document.getElementById('title');
        titleInput.value = field.getValue() ? field.getValue() :
'';

        titleInput.addEventListener('change', function(e) {
            field.setValue(titleInput.value);
        });
    }
});
```

Validating System and Custom Field Values

Validating field values is optional. If you would like to validate values before setting values in the item or field, you can invoke their respective validate methods to check if

the value is valid. These validate methods work similar to the validation in out-of-the box content forms.

```
// validating name of the item
var nameField = document.getElementById('name-input');
nameField.value = itemProps.name ? itemProps.name : '';
nameField.addEventListener('change', function (e) {
    item.validateName(nameField.value).then(function (validation) {
        if(validation && validation.isValid) {
            // set name of the item
            item.setName(nameField.value);
        } else {
            // display name validation error
        }
    }).catch(function (error) {
        // handle error
    });
});

// validating a field value
var titleInput = document.getElementById('title');
titleInput.value = field.getValue() ? field.getValue() : '';
titleInput.addEventListener('change', function (e) {
    field.validate(titleInput.value).then(function (validation) {
        if(validation && validation.isValid) {
            field.setValue(titleInput.value);
        } else{
            //display field validation error
        }
    }).catch(function (error) {
        // handle error
    });
});
```

For multi-valued fields, when you want to validate values for a specific index, you can pass an additional parameter `{index: <index>}`.

```
field.validate(value, {index:2}).then(function (validation) {
// handle validation result
});
```

Validating Form

Content form SDK provides a mechanism to implement custom validation of the form by registering a validation callback function. This can be done via `sdk.registerFormValidation(callback)`. If such registration and callback function is present, then the callback function will be invoked from the interface when the form is saved. This callback function should handle form validation and return whether or not the form is valid.

```
var item;

function initForm(sdk) {
    item = sdk.getItem();
```

```
// validation callback registration
sdk.registerFormValidation(validateForm);

// ... rest of the form init code
}

function validateForm() {
  var isValid = true,
      message = '';
  if (!item.get().name) {
    isValid = false;
    message = 'Name is required'
  }
  return {
    isValid: isValid,
    message: message
  }
}
```

Previewing Assets from the Form

Content form SDK provides an option to preview other assets that the item references. This can be done by invoking `sdk.previewAsset({id: <assetId>})` and passing an id of the asset you want to preview. This opens up the asset in preview drawer.

```
sdk.previewAsset({id: <asset_id>});
```

Content Form SDK Reference

The Oracle Content Management content form SDK API `content-form-sdk-1.0.js` is the library that handles the communication between the custom form component and the interface. This has a global variable called `contentFormSDK`. Custom form implementation should invoke `contentFormSDK.init()` method and pass a callback function to begin the communication between the interface and the custom form.

- [Custom Content Form SDK Object](#)
- [Custom Content Form Type Object](#)
- [Custom Content Form Item Object](#)
- [Custom Content Form Field Object](#)
- [CustomEditor Object](#)

Custom Content Form SDK Object

The callback function passed to `contentFormSDK.init()` gets an instance of the SDK object that provides following methods:

Method	Parameter	Required	Returns	Usage
<code>sdk.getLocale()</code>	None	n/a	Returns the current locale of the interface.	<pre>var locale = sdk.getLocale();</pre>
<code>sdk.getItem()</code>	None	n/a	Returns the Item object.	<pre>var item = sdk.getItem();</pre>
<code>sdk.getType()</code>	None	n/a	Returns Type object of the content item's content type.	<pre>var type = sdk.getType();</pre>
<code>sdk.resize(size)</code>	Object with height and width.	no	Resizes the frame to specified size. When no size is specified, it will resize the iframe to the size of its body.	<pre>sdk.resize({ width: '80%', height: '80%' });</pre>
<code>sdk.registerFormValidation(callback)</code>	Function that handles the validation of the form.	yes	This callback will be invoked to validate the form when saving.	<pre>sdk.registerFormValidation(function(){ //handle form validation });</pre>
<code>sdk.previewAsset(options)</code>	options- Object with id. {id:<asset_id>}	yes	Opens up asset preview drawer	<pre>sdk.previewAsset({ id: '<asset_id>' });</pre>
<code>sdk.getRepositoryDefaultLanguage()</code>	None	n/a	Returns the default language of the current repository.	<pre>var lang = sdk.getRepositoryDefaultLanguage();</pre>

Method	Parameter	Required	Returns	Usage
<code>sdk.createAsset(options)</code>	<p>options- Object with type</p> <pre>{ type:<type_name>, header: { <header-button>: true false }, callBack: function() {} }</pre>	<p>type - optional ; header - optional ; callBack - optional ; the function is to be invoked when newly created asset is saved.</p>	<p>When the type is given, it opens up the create item drawer. When the type is not given, it opens up the dialog to choose a type. And after the type is chosen, it opens the create item drawer.</p> <p>When invoking <code>sdk.createAsset()</code>, if its response needs to be handled once, then pass resolve and reject callbacks to the Promise. When the Promise is fulfilled, it returns a newly created asset response, but returns error object containing the error response when rejected.</p> <pre>sdk.createAsset({ type: '<type-name>' }).then(function (response) { // handle response }).catch(function (error) { // handle error });</pre> <p>When invoking <code>sdk.createAsset()</code>, if its response needs to be handled every time when save happens in the drawer that creates new asset, then a <code>callBack</code> parameter should be passed.</p> <pre>sdk.createAsset({ type: '<type_name>', header: { save: true },</pre>	<pre>sdk.createAsset({ type: '<type- name>' }).then(function (response) { // handle response }).catch(function (error) { // handle error });</pre> <pre>sdk.createAsset({ type: '<type_name>', header: { save: true }, callBack: function(response){ // handle response } });</pre>

Method	Parameter	Required	Returns	Usage
			<pre> callback: function(response){ // this function will be // invoked every time save occurs } }); </pre>	<p>callback option may be passed if the save option in the header is set to true and the form invoking <code>sdk.createAsset()</code> needs to listen to save.</p>

Method	Parameter	Required	Returns	Usage
sdk.editAsset(options)	options- Object with id <pre> { id: <asset_id> header: { <header-button>: true false }, callBack: function() {} } </pre>	id - required; header - optional; callBack - optional , the function is to be invoked when asset being edited is saved.	<p>When the given asset is a content item, it opens up edit item drawer.</p> <p>When the given asset is a digital asset, it opens edit attributes drawer.</p> <p>When invoking <code>sdk.editAsset()</code>, if its response needs to be handled once, then pass resolve and reject callbacks to the Promise. When the Promise is fulfilled, it returns the edited asset's response; but returns error object containing the error response when rejected.</p> <p>When invoking <code>sdk.editAsset()</code>, if its response needs to be handled every time when save happens in the drawer that edits the asset, then a <code>callBack</code> parameter should be passed.</p>	<pre> sdk.editAsset({ id: '<asset_id>' }).then(function (response) { // handle response }).catch(function (error) { // handle error }); sdk.editAsset({ id: '<asset_id>', header: { save: true }, callBack: function(response){ // handle response } }); </pre>

Method	Parameter	Required	Returns	Usage
			<pre> // handle response } }); </pre> <p>callback option may be passed if the save option in the header is set to true and the form invoking <code>sdk.editAsset()</code> needs to listen to save.</p>	
<code>sdk.getRepositoryId()</code>	None	n/a	Returns Id of the current repository.	<pre> var currentRepo = sdk.getRepositoryId(); </pre>
<code>sdk.isMediaEditable(options)</code>	options- Object with id {id: <asset_id>}	id - required	Whether or not the media of the given digital asset is editable. { isEditable: true false, reason: '' <reason text when not editable> }	<pre> sdk.isMediaEditable({ id: '<asset_id>' }).then(function (response) { if(response && response.isEditable){ // try editing } }).catch(function (error) { //handle error }); </pre>
<code>sdk.editMedia(options)</code>	options- Object with id	id - required	Opens edit media drawer. When the digital asset's media is an image, opens image editor drawer. When digital asset's media is an advanced video, opens video editor drawer.	<pre> sdk.editMedia({ id: <asset_id> }).then(function (response) { // handle response }).catch(function (error) { // handle error }); </pre>

Method	Parameter	Required	Returns	Usage
sdk.isAssetEditable(options)	options- Object with id	id - required	Whether or not the asset is editable. { isEditable: true false, reason: '' <reason text when not editable> }	<pre> sdk.isAssetEditable({ id: '<asset_id>' }).then(function (response) { if(response && response.isEditable){ // try editing } }).catch(function (error) { //handle error }); </pre>
sdk.getDirection()	None	n/a	Returns the current direction of the UI. Either ltr or rtl	<pre> // when the form wants to handle and support // directionality let dir = sdk.getDirection(); if(dir){ document.querySelector("html").setAttribute("dir", dir); } </pre>
sdk.canCreateAsset(options)	options - object with type	type - required	Whether or not the asset of a given type can be created. { canCreate: true false, reason: '' <reason text when not able to create> }	<pre> sdk.canCreateAsset({ type: '<asset_type>' }).then(function (response) { if(response && response.canCreate){ // try creating } }).catch(function (error) { //handle error }); </pre>

SDK Constants

The custom content form SDK has the following field data type constants:

```

sdk.dataTypes.TEXT
sdk.dataTypes.LARGETEXT
sdk.dataTypes.REFERENCE
sdk.dataTypes.NUMBER

```



```

sdk.dataTypes.DECIMAL
sdk.dataTypes.BOOLEAN
sdk.dataTypes.DATETIME
sdk.dataTypes.JSON

```

Custom Content Form Type Object

`sdk.getType()` holds the data of the content type of the item .

Method	Parameter	Returns	U S a g e
<code>type.getSlug()</code>	None	Slug property of type definition properties node. <pre> "slug": { "enabled": false, "pattern": "-{name}" } </pre>	v a r s l u g = t y p e . g e t S l u g () ;

Method	Parameter	Returns	Usage
type.getGroups()	None	Groups property of type definition properties node. <pre> "groups": [{ "title": "Content Item Data Fields", "collapse": false }, { "title": "Blog Fields", "collapse": true }] </pre>	<pre> v a r g r o u p s = t y p e . g e t G r o u p s () ; </pre>

Custom Content Form Item Object

sdk.getItem() returns an Item object which holds item data.

Method	Parameter	Required	Returns	Usage
item.get()	None	n/a	Basic item properties.	<pre>var itemProperties = item.get();</pre>

Method	Parameter	Required	Returns	Usage
			<pre><latestVersion>, mimeType: <mimeType>, fileGroup: <fileGroup>, varSetId: <varSetId> }</pre>	
item.getLanguageOptions()	None	n/a	Method to return a list of language options applicable for creating new item. This method works for a new item only.	<pre>//when item is new build language dropdown if(item.isNew()){ var langOptions = item.getLanguageOptions(); }</pre>
item.isNew()	None	n/a	Method to specify whether or not the item is new.	<pre>var isNewItem = item.isNew();</pre>
item.getFields()	Object with height and width.	n/a	Method to return an array of Field objects.	<pre>var fields = item.getFields();</pre>
item.getFieldById(fieldId)	field ID string	yes	Method to return a Field object by field ID.	<pre>var field = item.getFieldById(<fieldId>);</pre>
item.getFieldByName(fieldName)	field name string	yes	Method to return a Field object by field name.	<pre>var field = item.getFieldByName(<fieldName>);</pre>

Method	Parameter	Req uire d	Returns	Usage
item.validateName(name)	name string	yes	Validates a given name and returns a Promise. When fulfilled returns validation object. // when valid { isValid: true, } // when invalid { isValid: false, errorMessageSumm ary: <error message> errorMessageDeta il: <detail error message> };	item.validateName(nameField.value).then(function (validation) { if (validation && validation.isValid) { // handle valid name } else { // display invalidation errors } }).catch(function (error) { // handle error });
item.validateDescription(d escription)	description string	yes	Validates given description and returns a Promise. When fulfilled returns validation object. // when valid { isValid: true, } // when invalid { isValid: false, errorMessageSumm ary: <error message> errorMessageDeta il: <detail error message> };	item.validateDescription(descField.value).then(function (validation) { if (validation && validation.isValid) { // handle valid description } else { //display invalidation errors } }).catch(function (error) { // handle error });

Method	Parameter	Req uire d	Returns	Usage
item.validateSlug(slug)	slug string	yes	Validates given slug and returns a Promise. When fulfilled returns validation object. // when valid { isValid: true, } // when invalid { isValid: false, errorMessageSumm ary: <error message> errorMessageDeta il: <detail error message> };	item.validateSlug(slugField.value).then(function (validation) { if (validation && validation.isValid) { // handle valid slug } else { // display invalidation errors } }).catch(function (error) { // handle error });
item.validateLanguage(language)	language String	yes	Validates given language and returns a Promise. When fulfilled returns validation object. // when valid { isValid: true, } // when invalid { isValid: false, errorMessageSumm ary: <error message> errorMessageDeta il: <detail error message> };	item.validateLanguage(language).then(function (validation) { if (validation && validation.isValid) { // handle valid } else { // display invalidation errors } }).catch(function (error) { // handle error });

Method	Parameter	Required	Returns	Usage
<code>item.setName(name,options)</code>	name - string options - object with {silent: true false}	name - yes options - optional	Sets item name. When options with <code>silent: true</code> property is passed, form doesn't become dirty.	<pre>// sets name and form becomes dirty item.setName(name); // sets name and form doesn't become dirty item.setName(name, {silent: true});</pre>
<code>item.setDescription(description,options)</code>	description - string options - object with {silent: true false}	description - yes options - optional	Sets item description. When options with <code>silent: true</code> property is passed, form doesn't become dirty.	<pre>item.setDescription(description); item.setDescription(description, {silent: true});</pre>
<code>item.setSlug(slug,options)</code>	description - string options - object with {silent: true false}	slug - yes options - optional	Sets slug value. When options with <code>silent: true</code> property is passed, form doesn't become dirty.	<pre>item.setSlug(slug); item.setSlug(slug, {silent: true});</pre>
<code>item.setLanguage(language,options)</code>	languageCode - string options - object with {silent: true false}	language - yes options - optional	Sets language. Works for new items only. When options with <code>silent: true</code> property is passed, form doesn't become dirty.	<pre>item.setLanguage('en-US'); item.setLanguage('en-US', {silent: true});</pre>
<code>item.setTranslatable(isTranslatable,options)</code>	isTranslatable - boolean options - object with {silent: true false}	setTranslatable - yes options - optional	Sets whether or not the item is translatable. Will work only if the item is a new master. When options with <code>silent: true</code> property is passed, form doesn't become dirty.	<pre>item.setTranslatable(isTranslatable); item.setTranslatable(isTranslatable, {silent: true});</pre>
<code>item.on(event, handler)</code>	event - string handler - function that handles the event	yes	Currently only 'update' event is supported.	<pre>item.on('update', function(value) { // called if the item values change in the UI // update item properties with latest value itemProps = value; });</pre>

Method	Parameter	Required	Returns	Usage
<code>item.getChannels()</code>	None	n/a	Returns a Promise. When fulfilled the channels of the item are returned.	<pre>item.getChannels().then(function (data) { console.log('Channels:', data); });</pre>
<code>item.getCollections()</code>	None	n/a	Returns a Promise. When fulfilled the collections of the item are returned.	<pre>item.getCollections().then(function (data) { console.log('Collections :', data); });</pre>
<code>item.getTags()</code>	None	n/a	Returns a Promise. When fulfilled the tags of the item are returned.	<pre>item.getTags().then(function (data) { console.log('Tags :', data); });</pre>
<code>item.getTaxonomies()</code>	None	n/a	Returns a Promise. When fulfilled the taxonomies of the item are returned.	<pre>item.getTaxonomies().then(function (data) { console.log('Taxonomies :', data); });</pre>
<code>item.getVersionInfo()</code>	None	n/a	Returns a Promise. When fulfilled the version information of the item is returned.	<pre>item.getVersionInfo().then(function (data) { console.log('Version info :', data); });</pre>
<code>item.getPublishInfo()</code>	None	n/a	Returns a Promise. When fulfilled the publish information of the item is returned.	<pre>item.getPublishInfo().then(function (data) { console.log('Publish info :', data); });</pre>

Method	Parameter	Required	Returns	Usage
<code>item.getPublishedChannels()</code>	None	n/a	Returns a Promise. When fulfilled the published channels of the item are returned.	<pre>item.getPublishedChannels() .then(function (data) { console.log('PublishedChannels :', data); });</pre>
<code>item.addChannels(channels, options)</code>	channels - Array of channel objects options - Object with {silent: true false}	channels - yes options - optional	Adds given channels to the item.	<pre>item.addChannels([{id: '<channel1>', ...}, {id: '<channel2>', ...}]); item.addChannels([{id: '<channel1>', ...}, {id: '<channel2>', ...}, {silent: true}]);</pre>
<code>item.removeChannels(channels, options)</code>	channels - Array of channel objects options - Object with {silent: true false}	channels - yes options - optional	Removes given channels from the item	<pre>item.removeChannels([{id: '<channel1>', ...}, {id: '<channel2>', ...}]); item.removeChannels([{id: '<channel1>', ...}, {id: '<channel2>', ...}, {silent: true}]);</pre>
<code>item.addTags(tags, options)</code>	tags - Array of tag objects options - Object with {silent: true false}	tags - yes options - optional	Adds given tags to the item	<pre>item.addTags([{name: 'tag1'}, {id: 'tag2'}]); item.addTags([{name: 'tag1'}, {id: 'tag2'}, {silent: true}]);</pre>
<code>item.removeTags(tags, options)</code>	tags - Array of tag objects options - Object with {silent: true false}	tags - yes options - optional	Removes given tags from the item	<pre>item.removeTags([{name: 'tag1'}, {id: 'tag2'}]); item.removeTags([{name: 'tag1'}, {id: 'tag2'}, {silent: true}]);</pre>

Method	Parameter	Required	Returns	Usage
item.addCollections(collections, options)	collections - Array of collection objects options - Object with {silent: true false}	collections - yes options - optional	Adds given collections to the item	<pre>item.addCollections([[id: '<collection1>', ...], {id: '<collection2>', ...}]); item.addCollections([[id: '<collection1>', ...], {id: '<collection2>', ...}], {silent: true});</pre>
item.removeCollections(collections, options)	collections - Array of collection objects options - Object with {silent: true false}	collections - yes options - optional	Removes given collections from the item	<pre>item.removeCollections([[id: '<collection1>', ...], {id: '<collection2>', ...}]); item.removeCollections([[id: '<collection1>', ...], {id: '<collection2>', ...}], {silent: true});</pre>
item.addCategories(taxonomy, categories, options)	taxonomy - taxonomy under which categories fall categories - Array of category objects options - Object with {silent: true false}	taxonomy - yes categories - yes options - optional	Adds given categories under given taxonomy to the item	<pre>item.addCategories(<taxonomy>, [{id: '<category1>', ...}, {id: '<category2>', ...}]); item.addCategories(<taxonomy>, [{id: '<category1>', ...}, {id: '<category2>', ...}], {silent: true});</pre>
item.removeCategories(taxonomy, categories, options)	taxonomy - taxonomy under which categories fall categories - Array of category objects options - Object with {silent: true false}	taxonomy - yes categories - yes options - optional	Removes given categories under given taxonomy from the item	<pre>item.removeCategories(<taxonomy>, [{id: '<category1>', ...}, {id: '<category2>', ...}]); item.removeCategories(<taxonomy>, [{id: '<category1>', ...}, {id: '<category2>', ...}], {silent: true});</pre>

Method	Parameter	Req uire d	Returns	Usage
item.getFormOptions()	None	n/a	Returns form options	<pre>var formOptions = item.getFormOptions(); supportsSetName: true false, supportsSetDescripti on: true false, supportsSetLanguage: true false, supportsSetTranslata ble: true false, supportsSetSlug: true false, supportsSetMetaData: true false, supportsSetNativeFil e: true false, supportsRequiredVali dation: true false, placement: sidebar drawer, mixedValueFields: [<field_name1>, <field_name2>..] }</pre>

Method	Parameter	Req uire d	Returns	Usage
item.getNativeFileOptions()	None	n/a	Returns native file options. This is an array of possible options to upload native file. First item in the array allows to upload the file from computer, followed by options that allow to use the file from existing translations.	<pre>var nativeFileOptions = item.getNativeFileOptions();</pre> <pre>[{ action: "add-from-computer", label: "Add from this computer" value: "add-from-computer" }, { action: "add-from-documents", label: "Add from Documents", value: "add-from-documents" }, { action: "add-from-translation" label: "Use <Language Name> Master" languageIsMaster: true value: <id of master> }, { action: "add-from-translation" label: "Use <Language Name>" languageIsMaster: false value: <id of the translation> }...]</pre>

Method	Parameter	Required	Returns	Usage
item.validateNativeFile()	file - instance of File object	yes	Validates given file object and returns a Promise. When fulfilled returns validation object. Validation is done to validate the file type. // when valid { isValid: true, } // when invalid { isValid: false, } errorMessageSummary: <error message> errorMessageDetail: <detail error message> };	This method should be used to validate the chosen file when the 'add-from-computer' option is selected to pick a native file. item.validateNativeFile(file).then(function (validation) { if (validation && validation.isValid) { // handle valid } else { // display invalidation errors } }).catch(function (error) { // handle error });
item.setNativeFile(file, options)	file - File object options - Object with {silent: true false}	no	Sets the given file	This method should be used to set the chosen file when the 'add-from-computer' option is selected to pick a native file. item.setNativeFile(file); item.setNativeFile(file, {silent: true});
item.setSourceId(sourceId, options)	sourceId - id of the existing translation options - Object with {silent: true false}	sourceId - yes options - optional	Sets the given sourceId	This method should be used to set the sourceId when the 'add-from-translation' option is chosen. Selected translation's id should be passed as sourceId. item.setSourceId(<id>); item.setSourceId(<id>, {silent: true});

Method	Parameter	Req uire d	Returns	Usage
item.openDocumentPicker()	None	n/a	Opens the document picker drawer and returns a Promise. When fulfilled returns an object containing id and name of the selected document.	<pre>item.openDocumentPicker().then(function (doc) { console.log('Selected document:', doc); }).catch(function (error) { // handle error });</pre>
item.validateDocument(doc)	doc - document object { id: <docId> name: <docName> }	yes	Validates given document object and returns a Promise. When fulfilled returns validation object. Validation is done to validate the extension of the document name.	<p>This method typically used to validate the chosen document when the 'add-from-documents' option is selected and when the document is chosen from the document picker.</p> <pre>item.validateDocument(doc).then(function (validation) { if (validation && validation.isValid) { // handle valid } else { // display invalidation errors } }).catch(function (error) { // handle error });</pre> <pre>// when valid { isValid: true, } // when invalid { isValid: false, errorMessageSummary: <error message> errorMessageDetail: <detail error message> };</pre>

Method	Parameter	Required	Returns	Usage
item.setDocument(doc, options)	doc - document object { id: <docId> name: <docName> } options - Object with {silent: true false}	yes doc - yes options - optional	Sets the given document.	This method typically used to set the document when the given document needs to be used for separate native file. item.setDocument(file); item.setDocument(file, {silent: true});

Form Options

The following form options are available for the custom content form item object.

Table 8-1 Form Options

Property	Description
supportSetsName	Whether or not the form supports setting name. Typically this value is false when adding or editing digital asset attributes. When this value is false, calling <code>item.setName()</code> will throw an error.
supportsSetDescription	Whether or not the form supports setting description. Typically this value is false when adding or editing digital asset attributes. When this value is false, calling <code>item.setDescription()</code> will throw an error.
supportsSetLanguage	Whether or not the form supports setting language. Typically this value is false when adding or editing digital asset attributes
supportsSetTranslatable	Whether or not the form supports setting translatable. Typically this value is false when adding or editing digital asset attributes, and adding or editing content items in a business repository. When this value is false, calling <code>item.setTranslatable()</code> will throw an error.

Table 8-1 (Cont.) Form Options

Property	Description
supportsSetSlug	<p>Whether or not the form supports setting slug. Typically this value is false in following scenarios:</p> <ul style="list-style-type: none"> • While adding multiple digital assets together to the repository • While adding/editing digital assets and content items in the business repository • While converting digital assets of seeded type to a custom type <p>When this value is false, calling</p> <pre>item.setSlug()</pre> <p>will throw an error.</p>
supportsSetMetaData	<p>Whether or not the form supports setting metadata. Typically this value is false while adding digital assets to the repository.</p> <p>When this value is false, calling following methods of item will throw an error.</p> <ul style="list-style-type: none"> • <code>addCollections()</code> • <code>removeCollections()</code> • <code>addChannels()</code> • <code>removeChannels()</code> • <code>addTags()</code> • <code>removeTags()</code> • <code>addCategories()</code> • <code>removeCategories()</code>
supportsRequiredValidation	<p>Whether or not the form supports required validation. Typically this value is false while adding digital assets to the repository.</p>
placement	<p>Whether the form is displayed in the</p> <pre>sidebar</pre> <p>or</p> <pre>drawer</pre> <p>.</p>

Table 8-1 (Cont.) Form Options

Property	Description
<code>mixedValueFields</code>	An array of field names that has mixed values while editing attributes across multiple selected assets when adding assets to repository.
<code>supportsSetNativeFile</code>	Whether or not the form supports adding a native file from the form for custom digital asset translations. Typically this value is true for new translations of custom digital assets whose Asset Type's file field is not inherit from master. In all other cases it is false.

Custom Content Form Field Object

`item.getFields()` returns the array of Field object. The Field object holds field data.

Method	Parameter	Required	Returns	Usage
field.getDefinition()	None	n/a	Returns field definition.	<pre>var fieldDefn = field.getDefinition();</pre> <pre>{ "id": "<id>", "name": "<name>", "description": "<description>", "datatype": "<datatype>", "required": true false, "valuecount": "single list", "properties": { "caas- translation": { "inheritFromMaster": false, "note": "", "translate": true } }, "settings": { // field settings } }</pre> <p>Settings property holds the field setting set for the field. Refer to Field Settings for more detail.</p>
field.getValue()	None	n/a	Returns field value. Refer to Field Types and Values for details.	<pre>var fieldValue = field.getValue();</pre>
field.getValueAt(index)	index Non-negative number	yes	Returns field value at specified index. Works for multi-valued fields only.	<pre>var value = field.getValueAt(2);</pre>

Method	Parameter	Required	Returns	Usage
field.validate(value, options)	value - value of the field options - optional	yes	Validates given field value and returns a Promise. When fulfills returns validation object. // when valid { isValid: true, } // when invalid { isValid: false, errorMessageSummary: <error message> errorMessageDetail: <detail error message> };	field.validate(fieldValue).then(function (validation) { if (validation && validation.isValid) { // handle valid field value } else { //display field validation error } }).catch(function (error) { // handle error });
field.setValue(value, options)	value - see section on field types and values options - object with {silent: true false}	no options - optional	Sets given value for the field. When options with silent: true property is passed, form doesn't become dirty.	field.setValue(fieldValue); field.setValue(fieldValue, {silent: true});
field.setValueAt(index, value, options)	index - Non-negative number options - object with {silent: true false}	index - yes options - optional	Sets given value at the specified index for the field. Works only for multi-valued fields. When options with silent: true property is passed, form doesn't become dirty.	field.setValueAt(1, fieldValue); field.setValueAt(1, fieldValue, {silent: true});
field.removeValueAt(index, options)	index - Non-negative number options - object with {silent: true false}	index - yes options - optional	Removes value at the specified index. Works only for multi-valued fields. When options with silent: true property is passed, form doesn't become dirty.	field.removeValueAt(1); field.removeValueAt(1, {silent: true});

Method	Parameter	Required	Returns	Usage
field.openAssetPicker(options)	options - parameters to be sent to asset picker. Refer to options of mediaPicker and referencePicker sections in Embed UI API V2 for Oracle Content Management , a JavaScript API options - Object	yes	Opens the asset picker and returns a Promise. When fulfilled returns an object containing id, name and type of the selected asset. <pre>{ id: <asset_id>, type: <asset_type>, name: <asset_name> }</pre>	<pre>field.openAssetPicker().then(function (data) { var newValue = { id: data.id, type: data.type, name: data.name }; field.setValue(newValue); }).catch(function (error) { // handle error });</pre>
field.on(event, handler)	event - string handler - function that handles the event	yes	Currently only 'update' event is supported.	<pre>field.on('update', function(value){ // called if the field changes in the UI });</pre>

Method	Parameter	Required	Returns	Usage
field.createCustomEditor(editorName, options)	<p>editorName - name of the custom field editor</p> <p>options - Object</p> <p>Can have index, width and height.</p> <p>index: If the field is multi-valued and the editor is needed to display field value for a specific index, then index can be passed.</p> <p>width and height: This method returns a CustomEditor object with the given custom field editor rendered in an iframe. If the iframe containing the field editor needs to be in a certain initial size, then the width and height parameters can be passed. Note: custom field editors can have their own fixed sizes. They might also trigger to adjust the containing frame's size.</p> <pre>{ index: <index>, width: <width>, height: <height> }</pre>	<p>editorName - yes</p> <p>options - optional</p>	Returns CustomEditor object	<pre>var fieldDefn = field.getDefinition(), isCustomEditor = fieldDefn.settings.caas.editor.hasOwnProperty('isCustom') ? fieldDefn.settings.caas.editor.hasOwnProperty('isCustom') : false; if(isCustomEditor){ var customFieldEditor = fieldDefn.settings.caas.editor.options.name; if(customFieldEditor){ // create a dom element var container = document.createElement('div'); document.body.appendChild(container); //create a custom editor for the field var customEditor = field.createCustomEditor(customFieldEditor); //add the custom editor's frame to the dom element container.appendChild(customEditor.getFrame()); //listen to custom editor's change event customEditor.on('change', function (value) { //set the updated value to the field field.setValue(value); }); }</pre>

Field Types and Values

Value parameter passed to `field.setValue()` or the value obtained from `field.getValue()` varies depending on the field data type and value count.

Datatype	Value count	Value	Sample value
text	single	String	"my text"
	list	Array of strings	["foo", "bar"]
largetext	single	String	"this is large text"
	list	Array of strings	["this is large text1", "this is large text2"]
reference	single	Object with type, ID and name	{ "id": "CONTCB9A907A21E14B73A678CF39CF14A0FC", "type": "DigitalAsset", "name": "bird.jpg" }
	list	Array of objects with type, ID and name	[{ "id": "CONT58729B8B480F45E891D584736AC44CBA", "type": "DigitalAsset", "name": "cheetah.jpg" }, { "id": "CONT393EBEDA72C84822A4C66D08CF2E4DBC", "type": "DigitalAsset", "name": "tiger.jpg" }]

Datatype	Value count	Value	Sample value
number	single	integer	200
decimal	single	decimal	245.25
boolean	single	true false	true
datetime	single	Object with value and timezone where value is the date string with format yyyy-MM-dd'T'HH:mm:ss.SSS+/-HH:mm and time zone is the timezone string	{ "value": "2020-08-21T03:20:00.000-04:00", "timezone": "America/New_York" }
json	single	json string	{ "name": "John", "age": 20, "gender": "Male" }

Field Settings

Settings property holds the information about the settings of the field including editor name and any field specific validators. Following is a sample settings property for a multi-valued text data type with multi-select SelectBox editor. Editor name set in the field definition can be obtained from `settings.caas.editor.name`. Available editors associated with field types are listed under [Field Types and Available Editors](#).

```
"settings": {
  "groupIndex": 3,
  "caas": {
    "description": "Fruits list",
    "valuecountRange": {
      "min": 1,
      "max": 3
    },
  },
  "customValidators": [],
  "editor": {
    "name": "multi-selectbox",
    "options": {
      "multiple": true,
      "valueOptions": [
```

```

        {
            "value": "apple",
            "label": "apple"
        },
        {
            "value": "banana",
            "label": "banana"
        },
        {
            "value": "orange",
            "label": "orange"
        }
    ]
}
}
}
}

```

Field Types and Available Editors

Data Type	Single	List
text	textbox textarea radiobuttonset single-selectbox single-selectbox-rest	textbox textarea checkboxset multi-selectbox multi-selectbox-rest tagcloud
largetext	textarea rich-text-editor markdown-editor	textarea rich-text-editor markdown-editor
number	number-inc-dec radiobuttonset single-selectbox	n/a
decimal	number-inc-dec radiobuttonset single-selectbox	n/a
boolean	boolean-switch boolean-checkbox	n/a
datetime	datepicker datetimestpicker datetimestpickerz	n/a
reference	mediapicker (in case of media) itempicker	mediapicker (in case of media) itempicker
json	json-form json-textarea	n/a

CustomEditor Object

`field.createCustomEditor()` returns the CustomEditor object. The Field object holds field data.

Method	Parameter	Required	Returns	Usage
<code>getFrame()</code>	None	n/a	The iframe element with src set to the custom field editor url.	<pre>var customEditor = field.createCustomEditor(c ustomFieldEditor); //get custom editor's frame var editorFrame = customEditor.getFrame();</pre>
<code>setDisabled(disable)</code> Note: this function should be invoked after the custom editor is ready.	true false	yes	Invokes the custom field editor's disable callback function registered via <code>registerDisable()</code> function if such exists for the custom field editor.	<pre>customEditor.on('editorRea dy', function () { //to disable the custom field editor customEditor.setDisabled(t rue); });</pre>
<code>getValue()</code>	None	n/a	Returns the value in the custom field editor.	<code>customEditor.getValue();</code>

Method	Parameter	Req uire d	Returns	Usage
validate()	None	n/a	<p>Invokes the custom field editor's validate callback function registered via setValidation() function if such exists for the custom field editor. Returns a Promise. When fulfilled returns validation object.</p> <pre>// when valid { isValid: true, } // when invalid { isValid: false, errorMessageSumm ary: <error message> errorMessageDeta il: <detail error message> };</pre>	<pre>customEditor.validate().th en(function (validation) { if (validation && validation.isValid) { // handle valid field value } else { //display validation error } }).catch(function (error) { // handle error });</pre>
resizeEditorFrame(size)	Object with height and width	yes	Resizes the frame containing the custom editor to specified size.	<pre>customEditor.resizeEditorF rame({ width: '600px', height: '250px' });</pre>

Method	Parameter	Req uire d	Returns	Usage
on(event, handler)	event - string handler - function that handles the event	yes	Supported events: <ul style="list-style-type: none"> • editorReady • change • editorResized 	<pre> customEditor.on('editorReady', function () { // if the editor needs any disabling, it can happen here }); customEditor.on('editorResized', function (data) { // if any dom elements needs to be adjusted }); // look for change even to grab the editor's changed value customEditor.on('change', function (value) { // validate field value if necessary field.validate(value).then(function (validation) { if (validation && validation.isValid) { // handle valid field value } else { //display field validation error } }).catch(function (error) { // handle error }); }); </pre>

Sample Custom Form

Oracle Content Management provides a sample custom form called `Starter-Blog-Post-Form` for use with the `Starter-Blog-Post` content type. This sample demonstrates the use of the content form SDK API to render, validate and set values for fields of item type `Starter-Blog-Post`.

The `Starter-Blog-Post-Form` sample is available in the Content Toolkit, which is available from [GitHub](#).

Get Custom Form Sample

To get the custom form sample, you must install and configure Content Toolkit if it is not yet installed.

<https://github.com/oracle/content-and-experience-toolkit/blob/master/sites/README.MD>

Once Content Toolkit is installed, you must use it to create a site template and upload the template to the Oracle Content Management server.

1. Create a site template.

```
cec create-template BlogTemplate -f BlogTemplate
```

This template includes content type `Starter-Blog-Post` and content form `Starter-Blog-Post-Form`.

2. Upload the template to an Oracle Content Management server.

```
cec upload-template BlogTemplate -s <registered server name>
```

Now when you use this template to create a site and create or edit an item of type `Starter-Blog-Post`, the custom content form sample `Starter-Blog-Post-Form` will be used.

Add OCM Image Picker and Link Dialog Plug-ins for Rich Text Editor

Oracle Content Management (OCM) provides image picker and link dialog features in the form of plug-ins for rich text editor, TinyMCE.



Note:

You may see OCM being referred to as Oracle Content Experience (OCE), the legacy name, in the code snippets.

To use TinyMCE for displaying large text field in a custom content form and have image and link functionality similar to a standard form, add these external plugins with some additional configuration. The plug-ins are compatible with TinyMCE version 6.3.1.

To add and configure the plug-ins:

1. Load `tinymce` from CDN.
Create a TinyMCE account and register with `tinymce` to obtain the `api-key` and replace the `<no-api-key>` with your `api-key` in the URL.

```
<!-- load TINYMCE from CDN-->  
<!-- obtain your own api-key for tinymce and replace the 'no-api-key'  
part of the url with your api-key-->  
<!-- Without actual api-key, tinymce editor will load with message
```

```
prompting to register-->
<script src="https://cdn.tiny.cloud/1/no-api-key/tinymce/6.3.1/
tinymce.min.js" referrerpolicy="origin"></script>
```

 **Note:**

Loading `tinymce` with the `no-api-key` option will load the TinyMCE editor with a prompt asking to register the domain.

2. Initialize the `tinymce` instance for the `Field` that is of type `largetext` and set the `Field` object as value in `editor.aceItemField`

```
tinymce.init({
  selector: '<dom-element-selector>',
  // list of plugins needed
  plugins: ['<plugin1>', '<plugin2>'],
  //load ace link and image plugins as external plugins
  external_plugins: {
    assetimage: '/documents/static/gemini/api/tinymce-plugins/
assetimage.js',
    assetlink: '/documents/static/gemini/api/tinymce-plugins/
assetlink.js',
  },
  // this property is important for the ace link plugin to work properly
  extended_valid_elements: 'a[*]',
  // other tinymce options
  // ....
  setup: function (editor) {
    editor = editor;
    // Note: set the current field in the editor
    // this must be set for the ace link and image plugins to work
    // this must be set for every tinymce editor instance.
    editor.aceItemField = field;
  }
});
```

9

Develop Translation Connectors for Language Service Providers

Oracle Content Management provides a connector framework for developers to develop their custom translation connectors. The connector framework is extensible, and a developer or partner can build a translation connector to work against any Language Service Provider (LSP).



Note:

Translation connectors aren't supported in private instances.

For translation by an LSP, Oracle Content Management converts a translation zip file into the format expected by the vendor and then packages up the response from the vendor into a zip file that Oracle Content Management can consume.

A custom translation connector needs to implement the following artifacts:

- REST interfaces for defining configuration, providing authorization, and translating content.
- Logic to create and manage a translation job, accept a translation job zip file, and return a translated version of the translation zip file

Oracle Content Management provides a sample translation connector implementation that runs against an included mock server to get you started.

Oracle Content Management provides the following tools for site and asset translations:

- A translation connector framework, a translation SDK, and command-line interface (CLI) utilities in Content Toolkit
- An administration web interface for creating a custom translation connector
- A translation job web interface that lets contributors submit assets or sites for translation by an LSP and import translations into Oracle Content Management



Note:

This chapter does not provide details on how to deploy or manage your connector. The model you choose for deployment will depend on the technology you use to implement the connector and the server you choose to deploy the connector.

The following topics describe how to develop custom translation connectors for Oracle Content Management and to translate Oracle Content Management sites and assets using LSPs:

- [Overview of the Translation Connector Framework](#)

- [Request a Lingotek Trial Connector for Content Translation](#)
- [Enable a Lingotek LSP Translation Connector](#)
- [Delete a Lingotek LSP Translation Connector](#)
- [Register Multiple Lingotek Connectors](#)
- [Add Custom Locales to a Lingotek Translation Connector](#)
- [Translate Native Files in Assets](#)
- [Build a New Translation Connector](#)
- [Configure and Register a Translation Connector](#)
- [Translation Connector Execution Flow](#)
- [Sample Translation Connector Implementation](#)
- [Understand the Sample Translation Connector Source Code](#)

Overview of the Translation Connector Framework

The Oracle Content Management translation framework enables you to translate sites and assets using external Language Service Providers.

To help you build a new translation connector, Oracle Content Management provides the Translation Connector SDK.

The connector framework abstracts the functionality that is provided and implemented by end connectors. It does so by providing the REST API interface for configuring a translation connector and for calling translation connector functionality to download translation jobs from sites and upload the resulting translations back into Oracle Content Management.

Translation Connector SDK

The Translation Connector SDK for Oracle Content Management is a sample NodeJS implementation of the Translation Connector REST APIs. The sample accepts a translation job zip file, translates all the resources in the file, which includes metadata file of site pages and both metadata file and native file of assets, and then returns a new zip file containing all the translations.

The Translation Connector SDK is part of Content Toolkit, which is available from [GitHub](#). To help you build a new translation connector, this SDK contains a sample translation connector:

- JSDoc for the sample translation connector
- A NodeJS sample translation connector implementation of the REST interfaces
- A NodeJS mock Language Server Provider that the sample runs against

 **Note:**

The Translation Connector SDK requires the user to have access to a language service provider (LSP) to do the actual string translations. A mock LSP server is included in the SDK to mimic the responses from the LSP by simply prepending the targeted locales onto the strings.

This SDK consists of three main modules:

- **SampleConnector:** A sample translation connector that implements the required Oracle Content Management Translation Connector API.
- **SampleJobManager:** A file-system based sample job manager that maintains the state of the connector jobs while they are translated.
- **TranslationProviderInterface:** A set of APIs used to call the Language Service Provider to submit documents for translation and retrieve the translated documents.

The Sample Translation Connector code is split up into three areas:

- **/translation-connector:**
 - **/connector:** Implements the required Oracle Content Management Cloud Translation Connector APIs.
 - **/job-manager:** This sample implementation of a file-based persistence store unzips the translation job, translates it, and then zips up the translations.
 - **/provider:** Implements the Language Service Provider APIs to upload, translate, and download documents.

In addition, you can use the mock LSP for testing:

- **/mockserver:** A mock LSP, which simply adds the requested translation locale to all the strings.

Translation connectors are deployable packages that use Restful services for the remote stores implementing the REST interfaces (SPIs/SDK) defined by Oracle Content Management. They internally use cloud store native SDKs to connect to remote systems.

Translation Connector REST APIs

You can deploy and run translation connectors anywhere and implement them with any technology stack, as long as they can be called from Oracle Content Management through REST APIs.

The Translation Connector SDK includes the following API classes:

- ConnectorApi
- MockTranslationProvider
- PersistenceStoreInterface
- SampleBasicAuth
- SampleConnector
- SampleConnectorRouter
- SampleFileDownloader

- SampleFileImporter
- SampleFilePersistenceStore
- SampleFileTranslator
- SampleJobManager
- TranslationFilter
- TranslationProviderInterface

Your translation connector needs to support the translation connector REST APIs.

The following table describes the REST APIs for Oracle Content Management translation connectors.

API	Description
GET <code>http://host:port/connector/rest/api</code>	Returns the versions of the Translation Connector REST API that the connector supports; for example: ["v1"]
GET <code>http://host:port/connector/rest/api/v1/server</code>	Returns the connector configuration information; for example, required parameters that the user needs to pass to the connector.
POST <code>http://host:port/connector/rest/api/v1/job</code>	Creates a new job in the connector and returns a unique translated job.
GET <code>http://host:port/connector/rest/api/v1/job/{id}</code>	Returns status information about the specified job.
POST <code>http://host:port/connector/rest/api/v1/job/{id}/translate</code>	Sends an Oracle Content Management translation zip file for the job to be translated.
GET <code>http://host:port/connector/rest/api/v1/job/{id}/translation</code>	Returns the translated job in the Oracle Content Management translated zip file format
DELETE <code>http://host:port/connector/rest/api/v1/job/{id}</code>	Deletes the job from the connector (and from the LSP if required)

Request a Lingotek Trial Connector for Content Translation

You can open a trial account with the Lingotek language service provider from the Oracle Content Management administration web interface.

To request a Lingotek trial account for content translation:

1. Sign in to the Oracle Content Management web interface as an administrator.
2. Click **Integrations** in the **Administration** area of the navigation menu.
3. In the **Integrations** drop-down menu, choose **Translation Connectors**.
4. On the **Translation Connectors** page, click **Lingotek** to open the connector configuration.
5. Click **Additional Fields**.
On the **Lingotek Additional Fields** page, you will input some data to configure your Lingotek trial account: the bearer token provided by Oracle, the default value

Oracle for Community, the default value *Machine Translation* for Workflow. See steps below for details.

The screenshot shows the 'Lingotek Connector' configuration page in Oracle Content Management. The page has a sidebar with navigation icons and a main content area. The 'Additional Fields' tab is selected, showing a table for additional data to be provided to Lingotek.

General | **Additional Fields**

* Bearer Token
This is the authorization token. Authorize access to your Lingotek account or request a trial Lingotek account.

Community
The community to use for your Lingotek translation projects

* Workflow
The workflow profile to use for your Lingotek translation projects

Additional Data
Additional data to be provided to Lingotek

Lingotek Field	Display Name	Value	Read-Only	Optional	Delete
None	Enter display name	Enter default value	<input type="checkbox"/>	<input type="checkbox"/>	×

- Under the **Bearer Token** field, click **request a trial Lingotek account**. The **Lingotek Request Account** page opens.

Lingotek Translation Management System

Request Account

Thank you for your interest in creating an account with Lingotek. To create your account, please provide the following information.

Company

First Name

Last Name

Email

Note: This email address will be used as your Lingotek Username.

United States

New Password

Confirm Password

Your password must be at least 'Good' complexity

I agree to the Lingotek Conditions of Use policy

You will be redirected to the Lingotek login screen.

Cancel Request Account

Copyright © 2020 Lingotek. All rights reserved. 801.331.7777 Conditions of Use Privacy Policy

7. Replace the values on this page with your data:
 - Replace **Company** with the name of your company.
 - Replace **First Name**, **Last name**, and **Email** with yours. The Email address will be used as your Lingotek user name.
 - Choose your country, such as **United States**, from the drop-down field.
 - Enter a new password and confirm it.
8. Click **Conditions of Use** to view the policy, and then select the check box to agree with it.
9. Click the **Request Account** button.
You will be redirected to the Lingotek sign-in page. Enter your Lingotek user name (email address) and password, and then click **SIGN IN**.
10. Select your organization (**Oracle**), and then click **ALLOW** to authorize the account.
11. The **Lingotek Credentials** page opens, with credentials for you to copy and paste into custom fields for your trial account.

Lingotek Credentials

Credentials for configuring Lingotek connector in OCE.

Bearer Token:

Use this value in the Bearer Token field of the OCE translation connector

Workflow Id:

Use this value in the Workflow Id field of the OCE translation connector

Community Id:

Title: Oracle

Use the credentials on this page to configure a Lingotek translation connector in Oracle Content Management. The credentials include **Bearer Token** and **Workflow Id** values for Oracle Content Management translation connectors as well as your **Oracle Community Id**.

Copy the **Bearer Token** value from the **Lingotek Credentials** page and paste it in the **Bearer Token** field on the Lingotek **Additional Fields** page. This populates the rest of the custom fields on the page.

12. After you click **Save**, you can use your Lingotek trial account.

Enable a Lingotek LSP Translation Connector

The Lingotek language service provider (LSP) translation connector is pre-registered. After you authorize Oracle on your Lingotek LSP account and enable the Lingotek LSP connector in the Oracle Content Management web interface, you can use the connector for translations.

To enable the Lingotek LSP translation connector:

1. Sign in to the Oracle Content Management web interface as an administrator or developer.
2. Click **Integrations** in the **Administration** area of the navigation menu.
3. In the **Integrations** menu, choose **Translation Connectors**.
4. Click **Enable** next to the Lingotek connector.

After you enable the Lingotek LSP connector, it becomes available in the **Language Service Provider** menu item of the Oracle Content Management **Create Translation Job** web interface.

Delete a Lingotek LSP Translation Connector

Before you delete the connector, you need to disable the connector and remove the translation jobs referencing that connector.

To delete the Lingotek LSP translation connector:

1. Sign in to the Oracle Content Management web interface as an administrator or developer.
2. Click **Integrations** in the **Administration** area of the navigation menu.
3. In the **Integrations** menu, choose **Translation Connectors**.
4. Click **Disable** next to the already created translation connector.
5. Select the translation connector and click **Delete** in the action bar.
If there are translation jobs referencing the connector, then an error message will be displayed and the connector will not be deleted.

Register Multiple Lingotek Connectors

You can register and enable multiple Lingotek language service provider (LSP) translation connectors in an Oracle Content Management instance.

Configuration of a Lingotek translation connector requires values for three additional fields:

- **Bearer Token:** The sign-in for your Lingotek account. You can use the same bearer token for all of your Lingotek connectors or specify a different one for a connector.
- **Community:** The community to use for your translation projects. The default is **Oracle**.
- **Workflow:** The workflow profile to use for your Lingotek translation projects. The default is **Machine Translation**.

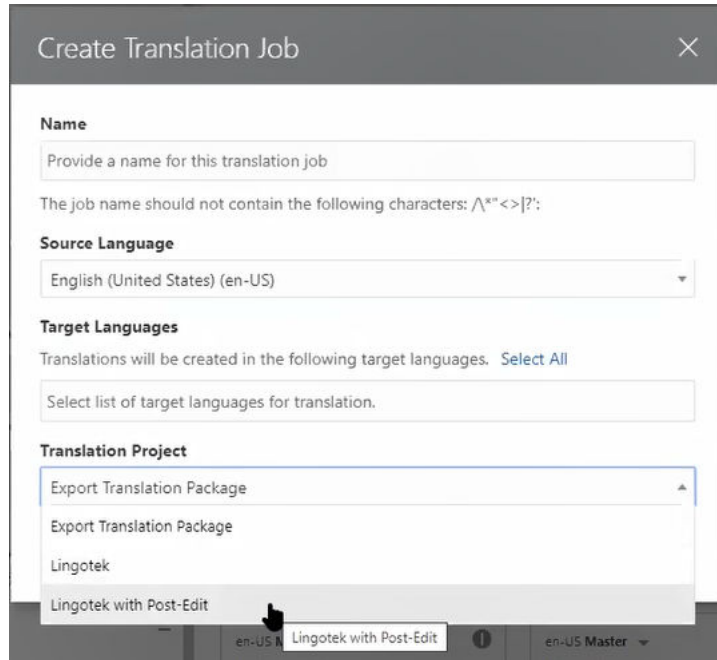
You can change any of these values. Use the Additional Data fields to define optional attributes of the connector.

The screenshot shows the 'Lingotek' configuration page in the Oracle Content Management interface. The page is divided into a left navigation menu and a main content area. The main content area has a header 'Lingotek' and two tabs: 'General' and 'Additional Fields'. Under the 'Additional Fields' tab, there are three fields:

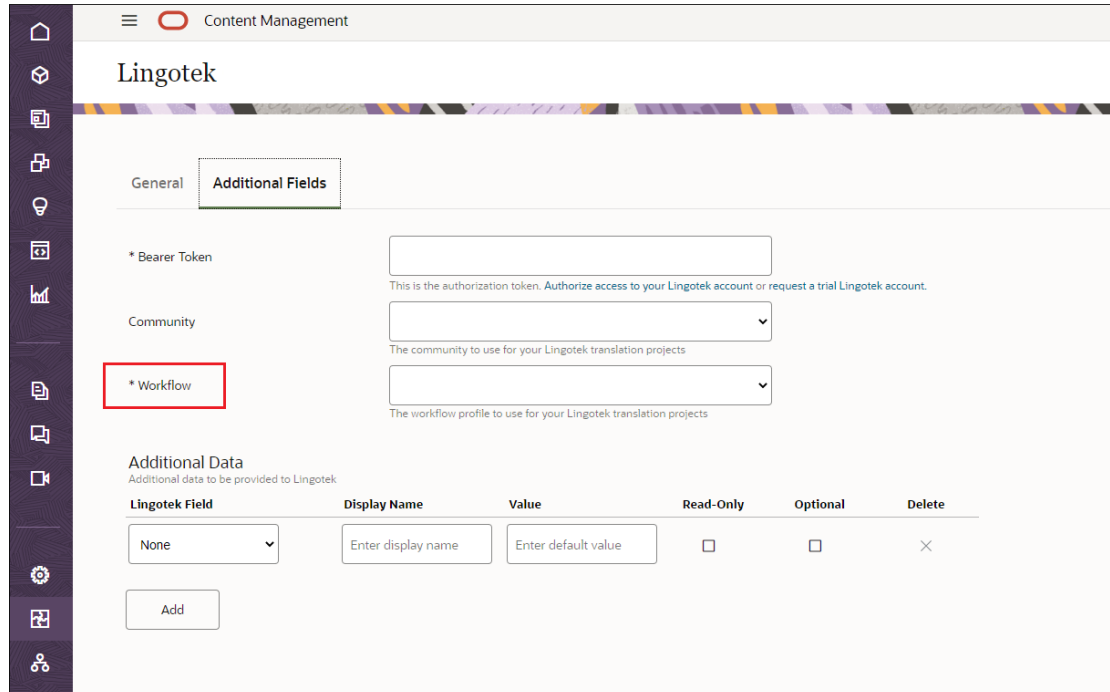
- * Bearer Token:** A text input field with a placeholder. Below it, a note says: "This is the authorization token. Authorize access to your Lingotek account or request a trial Lingotek account."
- * Workflow:** A dropdown menu with a downward arrow. Below it, a note says: "The workflow profile to use for your Lingotek translation projects"
- Community:** A dropdown menu with a downward arrow. Below it, a note says: "The community to use for your Lingotek translation projects"

 Below these fields is the 'Additional Data' section, which contains a table with the following columns: 'Lingotek Field', 'Display Name', 'Value', 'Read-Only', 'Optional', and 'Delete'. The table has one row with 'None' in the 'Lingotek Field' column, and input fields for 'Display Name' and 'Value'. There are checkboxes for 'Read-Only' and 'Optional', and a 'Delete' button with an 'x' icon. An 'Add' button is located below the table.

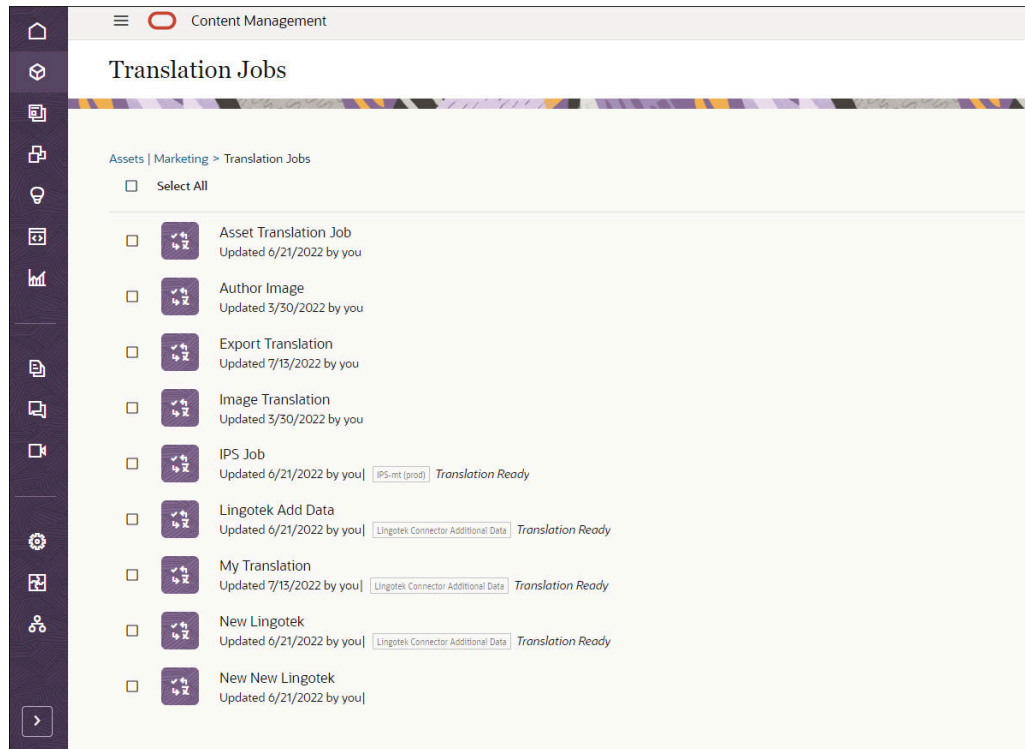
To create a translation job for an asset, you can select the asset in a repository and then choose **Translate** from the drop-down menu to open the **Create Translation Job** dialog.



The **Workflow** field shows workflows that you can use for your translation jobs. You can specify a workflow for a connector on the **Additional Fields** page when you create the connector.



The **Translation Jobs** page, under **Assets**, shows the status of jobs you have running on your Lingotek translation connectors.



If the workflow is different than **Machine Translation**, a translator edits or finishes the job, and you will need to approve each of the translator's changes before the job is 100% complete.

Add Custom Locales to a Lingotek Translation Connector

You can add custom languages to the list of languages known to Oracle Content Management and use them for managing asset translation and content delivery, including language fallback.

This image shows the fields to create a translation job, including Name, Source Language, Target Languages, and Translation Project.

Create Translation Job [X]

Name

 The job name should not contain the following characters: `^* <> [?]`:

Source Language
 English (en) [v]

Target Languages
 Translations will be created in the following target languages. [Select All](#)

Translation Project
 Export Translation Package [v]

Cancel Create

Oracle Content Management supports custom locales in the `<base or seeded language code>-<x>-<custom string>` format, where `<custom string>` is a dash-separated string of sub-strings up to 8 characters long.

A predefined list of languages (BCP-47), language codes (ISO-639-2), territory sub-tags (ISO-3166), and script sub-tags (ISO-15924) follows the BCP-47 standard (<https://tools.ietf.org/html/bcp47>). The names are restricted to the characters a-z, A-Z and 0-9 as stated in the BCP47 doc:

```
singleton = DIGIT ; 0-9
           / %x41-57 ; A - W
           / %x59-5A ; Y - Z
           / %x61-77 ; a - w
           / %x79-7A ; y - z
```

A content architect can add a custom locale to the list of languages available in Oracle Content Management as a Description and Code pair.

Description will be a locale name, such as English - APAC or Global French. Code will be a custom locale code, such as en-x-region-apac or fr-x-global. For example:

- en-x-custom-string OR ar-x-middle-east
- en-GB-x-custom-string-custom OR ar-SA-x-custom-string-custom-string.

With custom locales, you can manage content localization and language fallback in delivery.

You can create L10N policies with a default language. If the default language is set, an asset request will return translation to the default language if the language in the request is not found. If your company has a global presence, you can support asset request fallback based on a language hierarchy that is defined for each region where the content is consumed.

Oracle Content Management can support a URL path segment mapped to a standard or custom language for delivery to multilingual sites. The path segment can be a country code, language code, language name, or custom name.

Translate Native Files in Assets

For an asset with a native file, both the attributes and the native file can be translated by Lingotek. However, not all native file types are supported by Lingotek. For the list of supported file types, see [Supported File Types](#).

Build a New Translation Connector

To build a new translation connector as a RESTful service, a developer needs to implement the RESTful interfaces provided by the connector framework.

The Oracle Content Management Cloud connector framework has the following interfaces:

- **APIResource Interface:** This interface is the starting point of the translation connector. You simply return the version that's supported.
- **ServerResource Interface:** When an administrator or developer registers a translation connector in the Oracle Content Management (OCM) server, OCM calls this service to get the basic details about the connector server, such as the authorization type, custom fields, and properties for the translation connector:

- **Authorization types:**

- * **OAuth Client Credentials:** The translation connector supports Client Credentials (two-Legged OAuth).

To access Language Service Provider using [Client Credentials](#) in Identity Cloud Store (IDCS), (two-Legged OAuth), a token is requested for the client application and calls to the Language Service Provider APIs are made using this client application's token. The connector declares the custom fields required to obtain the token; for example, `Client ID`, `Client Secret`, and `Scope`. In addition, a custom field with fields ID `accessToken` must be declared. Every request made to the connector includes these custom fields in the request headers. In a new connector instance, the value of the `accessToken` field is empty. The connector should obtain an access token with the values of the other custom fields such as `Client ID`, `Client Secret`, and `Scope`. Once an access token is obtained, then the value should be set in the response header. The OCM connector framework saves the `accessToken` value in the response header. Subsequent requests for the connector will receive the `accessToken`.

To revoke the access token generated in the two-legged OAuth flow, select **Integrations** from the left navigation panel, select **Translation Connectors** in the dropdown menu. Select the checkbox corresponding to the translation connector, click the **Revoke Access Token** button.

 **Note:**

The **Revoke Access Token** button appears only when you've selected the checkbox corresponding to the connector, and the connector must be of type `OAuth Client Credentials`. Otherwise, the **Revoke Access Token** button will not appear.

- * **NO_AUTH:** The translation connector uses custom fields to specify fields such as `Bearer token` for authentication.
- **Custom fields and properties:** You can also use this interface to define custom fields for the translation connector. Custom properties are specific to a connector, so the connector framework can't provide such a properties list by itself. Every translation connector has its own requirements to connect to a remote store; for example, one translation connector might need just `Client ID` and `Client Secret`, while another might require `Client ID`, `Client Secret`, `App ID`, and so on. Each translation connector can provide a custom properties list to the Oracle Content Management server by **ServerResource**. If any of these properties need to be filled in by an administrator during configuration, the connector framework can surface them in the Administration web interface and pass them through as header parameters on each request to the translation connector.

This service will be called only once, at the time of registration.

- **TranslationResource Interface:** The connector framework uses this service to submit an Oracle Content Management translation job to the translation connector. The translation connector then calls the Language Service Provider to translate the zip file provided for the job and returns the translated zip.

Each of these interfaces is described in more detail, with examples of REST payloads, in the following sections:

- [REST Interfaces for Configuration and Authorization](#)
- [REST Interfaces for Creating Translation Jobs and Returning Translated Content](#)

REST Interfaces for Configuration and Authorization

A translation connector needs to implement the following REST APIs for defining the connector configuration and setting up authorization.

`/rest/api`

Implements `intradoc.connectorcommon.server.APIResource`
Here you return the latest version supported by the content connector.

GET `http://host:port/connector/rest/api`

```
[ "v1" ]
```

`/rest/api/v1/server`

Implements `intradoc.connectorcommon.server.ServerResource`

This returns information about the translation connector configuration, like the authentication type and custom fields it exposes.

GET `http://host:port/connector/rest/api/v1/server`

```
{
  "name": "",
  "nameLocalizations": [{
    "locale": "en_US",
    "localizedString": "Sample Connector"
  }]
}
```

```

    }],
    "version": "(1,0,0)",
    "about": "This is a sample connector.Copyright (c) 2019, Oracle
and/or its affiliates. All rights reserved.",
    "aboutLocalizations": [{
        "locale": "en_US",
        "localizedString": "This is a sample connector.Copyright (c)
2019, Oracle and/or its affiliates. All rights reserved."
    }],
    "authenticationType": "NO_AUTH",
    "pickerType": "",
    "enableMultiUserCopyBack": false,
    "maxUploadSize": 1073741824,
    "fields": [{
        "ID": "ProxyHost",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "HTTP Proxy Hostname",
        "labelLocalizations": [{
            "locale": "en",
            "localizedString": "HTTP Proxy Hostname"
        }],
        "description": "The HTTP proxy hostname, leave blank to
disable.",
        "descriptionLocalizations": [{
            "locale": "en",
            "localizedString": "The HTTP proxy hostname, leave blank
to disable."
        }],
        "required": false
    }, {
        "ID": "ProxyPort",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "HTTP Proxy Port",
        "labelLocalizations": [{
            "locale": "en",
            "localizedString": "HTTP Proxy Port"
        }],
        "description": "The HTTP proxy port number, leave blank to
default to port 80.",
        "descriptionLocalizations": [{
            "locale": "en",
            "localizedString": "The HTTP proxy port number, leave
blank to default to port 80."
        }],
        "required": false
    }, {
        "ID": "ProxyScheme",

```

```

        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "HTTP Proxy Scheme",
        "labelLocalizations": [{
            "locale": "en",
            "localizedString": "HTTP Proxy Scheme"
        }],
        "description": "The HTTP proxy scheme, leave blank to default to
http.",
        "descriptionLocalizations": [{
            "locale": "en",
            "localizedString": "The HTTP proxy scheme, leave blank to
default to http."
        }],
        "required": false
    }, {
        "ID": "BearerToken",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "Bearer Token",
        "labelLocalizations": [{
            "locale": "en",
            "localizedString": "Bearer Token"
        }],
        "description": null,
        "descriptionLocalizations": [],
        "required": true
    }, {
        "ID": "WorkflowId",
        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "Workflow Id",
        "labelLocalizations": [{
            "locale": "en",
            "localizedString": "Bearer Token"
        }],
        "description": null,
        "descriptionLocalizations": [],
        "required": true
    }],
    "supportedConnectorTypes": [],
    "proprietaryName": "",
    "serviceProviderName": "Sample APIs INC.",
    "nativeAppInfos": null
},

```

/rest/api/v1/authorization/authorizationURLsImplements `intradoc.connectorcommon.server.AuthorizationResource`

This is required only if the custom translation UI is to be configured. It adds a new field "Translation Job Editor" in the fields of the custom translation connector in Oracle Content Management.

```
{
  "ID": "CustomTranslationUI_name",
  "datatype": "STRING",
  "siteSettable": true,
  "userSettable": false,
  "connectorSettable": false,
  "authorizationURLParameter": false,
  "label": "Custom Translation UI",
  "labelLocalizations": [{
    "locale": "en",
    "localizedString": "Custom Translation UI"
  }],
  "description": null,
  "descriptionLocalizations": [],
  "required": false
},
{
  "ID": "CustomTranslationUI_id",
  "datatype": "STRING",
  "siteSettable": true,
  "userSettable": false,
  "connectorSettable": false,
  "authorizationURLParameter": false,
  "label": "Translation Job Editor",
  "labelLocalizations": [{
    "locale": "en",
    "localizedString": "Custom Translation UI Id"
  }],
  "description": null,
  "descriptionLocalizations": [],
  "required": false
},
```

The SDK exposes the following global object and properties that can be used in custom translation editor UI to find out the details about the translation job:

```
window.translationEditorSDK: {
  properties = {
    "connector": {
      "jobId": "string returned by the connector for the ID of the
job",
      "title": "string for the name of the job"
    },
    "translationJob": {
      "type": ['assets' | 'sites']
      "data": "object providing additional information about the
```

```

selected items based on job type"
    },
    "componentName": "string for the name of the translation editor
component",
    "locale": "string for the locale of the page"
  }
}

```

This is required only if the translation connector supports OAUTH_CLIENT_CREDENTIALS. It adds custom fields to obtain an access token and the required accessToken field.

```

{
  "ID": "ClientID",
  "datatype": "STRING",
  "siteSettable": true,
  "userSettable": false,
  "connectorSettable": false,
  "authorizationURLParameter": false,
  "label": "ClientID",
  "labelLocalizations": [{
    "locale": "en",
    "localizedString": "ClientID"
  }],
  "description": null,
  "descriptionLocalizations": [],
  "required": true
},
{
  "ID": "ClientSecret",
  "datatype": "STRING",
  "siteSettable": true,
  "userSettable": false,
  "connectorSettable": false,
  "authorizationURLParameter": false,
  "label": "ClientSecret",
  "labelLocalizations": [{
    "locale": "en",
    "localizedString": "ClientSecret"
  }],
  "description": null,
  "descriptionLocalizations": [],
  "required": true
},
{
  "ID": "Scope",
  "datatype": "STRING",
  "siteSettable": true,
  "userSettable": false,
  "connectorSettable": false,
  "authorizationURLParameter": false,
  "label": "Scope",
  "labelLocalizations": [{
    "locale": "en",

```

```

        "localizedString": "Scope"
    }],
    "description": null,
    "descriptionLocalizations": [],
    "required": true
},
{
    "ID": "accessToken",
    "datatype": "STRING",
    "siteSettable": false,
    "userSettable": false,
    "connectorSettable": true,
    "authorizationURLParameter": false,
    "label": "null",
    "labelLocalizations": [{
        "locale": "en",
        "localizedString": null
    }],
    "description": null,
    "descriptionLocalizations": [],
    "required": false
}

```

This is required if the translation connector supports Additional Data fields.

```

"additionalSettings": {
    "AdditionalDataFields": [
        {
            "id": "business_division",
            "name": "Business Division",
            "type": "String"
        },
        {
            "id": "email",
            "name": "Email",
            "type": "String"
        },
        {
            "id": "purchase_order",
            "name": "Purchase Order",
            "type": "String"
        },
        {
            "id": "finish_by",
            "name": "Finish By",
            "type": "Date"
        }
    ]
}
"fields": [{
    <<Other data>>
    {
        "ID": "AdditionalData",

```

```

        "datatype": "STRING",
        "siteSettable": true,
        "userSettable": false,
        "connectorSettable": false,
        "authorizationURLParameter": false,
        "label": "Additional Data",
        "labelLocalizations": [{
            "locale": "en",
            "localizedString": "Additional Data"
        }],
        "description": null,
        "descriptionLocalizations": [],
        "required": true
    },
}
<<Other fields>>
]

```

REST Interfaces for Creating Translation Jobs and Returning Translated Content

A translation connector needs to implement the following REST APIs for creating translation jobs, providing status about the connector translation jobs, and returning the translation results for each job.

/rest/api/v1/job

Implements `intradoc.connectorcommon.server.TranslationJobResource`
Given an OCE Translation Job ID, create and return a connector project ID for the translation job.

POST `http://host:port/connector/rest/api/v1/job`

Request Headers

```

X-CEC-ClientID:client-id
X-CEC-ProxyHost:proxy-host
X-CEC-ProxyPort:80
X-CEC-ProxyScheme:http
X-CEC-TranslationJob:translation-job-name

```

Request Body

```
{Zipfilecontent}
```

Response Headers

```

content-type application/zip
content-length {zip file size}

```

Response Body

```

{
    "properties": {
        "status": "IMPORTING",
        "id": "{{connectorJobId}}"
    }
}

```


/rest/api/v1/job/{id}/translation

Implements `intradoc.connectorcommon.server.TranslationJobResource`
Given a connector job id, return the translated zip file.

GET `http://host:port/connector/rest/api/v1/job/{id}/translation`

Request Headers

X-CEC-ClientID:client-id
X-CEC-ProxyHost:proxy-host
X-CEC-ProxyPort:80
X-CEC-ProxyScheme:http
X-CEC-TranslationConnectorJobId:connector-job-id

Request Body

Response Headers

content-type application/zip
content-size {zip file size}

Response Body

{streamed zip file}

/rest/api/v1/job/{id}

Implements `intradoc.connectorcommon.server.TranslationJobResource`
Given a connector job id, delete the connector job from the connector.

DELETE `http://host:port/connector/rest/api/v1/job/{id}`

Request Headers

X-CEC-ClientID:client-id
X-CEC-ProxyHost:proxy-host
X-CEC-ProxyPort:80
X-CEC-ProxyScheme:http
X-CEC-TranslationConnectorJobId:connector-job-id

Request Body

Response Headers

content-type application/json

Response Body

{Message noting successful deletion}

Configure and Register a Translation Connector

Once you have built your translation connector, you need to register it through the Oracle Content Management administration interface.

The minimum properties you are required to add to a translation connector follow:

- Translation connector name

- Translation connector service URL
- User name and password, if the preceding URL access requires it

Once a translation connector gets registered successfully and the translation connector service URL is reachable, the Oracle Content Management server will call a server service on the translation connector to get the custom properties and show those in the **Administration** web interface. An administrator or developer can add values for those properties (for example, `clientid` and `clientsecret` for OAuth flows or the user name and password for a Basic authorization central account).

Clicking **Save** saves all these properties with the connector framework.

You need to select the registered connector in the repository before you can use it in the rest of the UI. Once you select the connector in the repository, then any asset in that repository can use this connector to translate.

After you enable it, the translation connector becomes available in the **Language Service Provider** menu item of the Oracle Content Management **Create Translation Job** web interface.

Translation Connector Execution Flow

From the Oracle Content Management web interface **Assets** page of the **Sites** page, you can click **translate** and choose a translation connector as an option from the **Create Translation Job** web interface.

During creation of the translation job, the connector framework makes REST API calls to the remote translation connector to create a corresponding job in the connector. The zip file for the translation job is then submitted to the connector. Once the connector successfully receives the translation zip file, the job creation process is complete.

The user can now see the status of the job within the connector when viewing the list of translation jobs, or on the job detail page. Once the status moves to 100%, the user can ask Oracle Content Management to ingest the translated zip. Oracle Content Management will call the connector to retrieve the translated zip and ingest it.

Whenever a translation job is deleted that is using a translation connector, the translation connector is also told to delete the associated job in the connector.

Since a lot of the calls to the translation connector occur as a background process, the web interface will indicate only the percent complete or failure of the connector job. Refer to the diagnostic logs for any failures.

Translation Job Editor

A Translation Job Editor is a UI component that lets you edit translation jobs, manage and monitor the status of your translation job assets. The editor is where you communicate your translation job requirements to the translation connector.

 **Note:**

The editor is an optional component of a translation connector. On the Connector Settings' page, the option to add the Translation Job Editor is shown to you if you configured custom translation UI. Without the editor, the status of the job within the connector is available to you after the connector successfully completes the job creation process.

Oracle Content Management (OCM) comes with an out-of-box Translation Job Editor in the OCM web UI. The underlying code implementation is in `window.translationEditorSDK`, which can be downloaded through the command-line interface (CLI) utilities in Content Toolkit. You can modify the JavaScript code to customize the Translation Job Editor.

 **Note:**

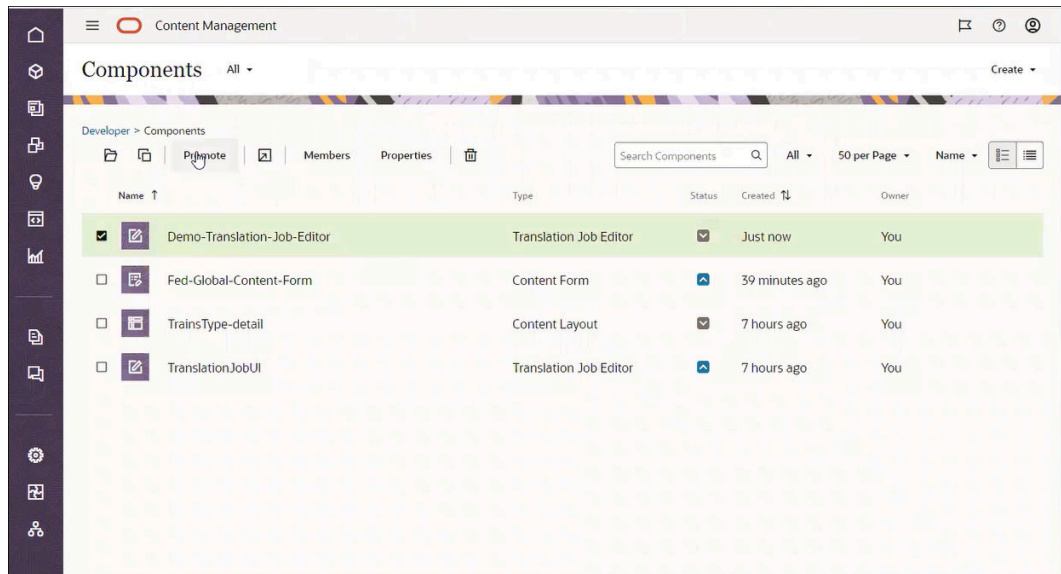
Customizing the editor in the provided code through the Content Toolkit will be available in the spring of 2023.

The following section covers the steps on how to create a Translation Job Editor in the OCM web UI. The high-level workflow is as follows:

- [Create the UI component called Translation Job Editor](#)
- [Configure the Component for the Custom Connector](#)
- [Create a Translation Job and Render the Editor](#)
- [Customize the Translation Job Editor](#)

Create the UI component called Translation Job Editor

1. In the OCM web UI, access the **Developer** tab > **Components** > **Create** > **Translation Job Editor**. Give it a name, for example: Demo-Translation-Job-Editor.
2. Promote the Translation Job Editor on the Components page. It's now available to be configured for a custom translation connector.



Configure the Component for the Custom Connector

1. • Download the [Translation Connector SDK](#), available in [OCM Toolkit](#)

Create a translation connector. See below for the toolkit command.

<code>cec create-translation-connector <name></code>	Creates translation connector <name>. [alias: ctc]
--	--

2. • Modify the `sampleConnectorResponses.js`

After creating the translation connector, locate the connector source code. Find the source code of the connector in `src/connectors/<name>`

Within that directory, look for the `sampleConnectorResponses.js` file and modify it according to the [documentation](#).

With the “CustomTranslationUI_name” and “CustomTranslationUI_id” fields defined in the custom connector implementation, the Connector Settings page will display the Translation Job Editor field.

- Start the translation connector. See below for the toolkit command. Then use the OCM web UI to configure the custom translation connector.

<code>cec start-translation-connector <name></code>	Starts translation connector <name>. [alias: stc] src/connectors/<name>
---	---

Note:

The translation connector must be started before you can configure the custom translation connector in the OCM web UI.

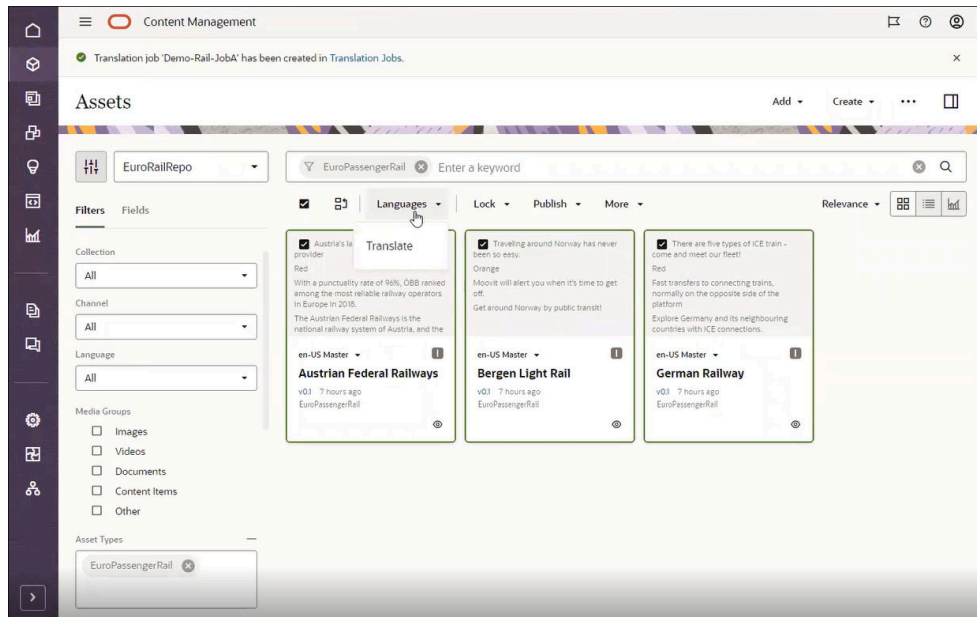
3. To configure a custom translation connector, access the **Integration** page and select **Translation Connectors** in the top dropdown menu. Select **Create > Custom Connector** to configure the connector. Enter the required fields: name, and connector service URL.

Go to the next page to fill in the required fields, plus the Translation Job Editor field that is now available. Select the Translation Job Editor component you've created and promoted. **Save** your data.

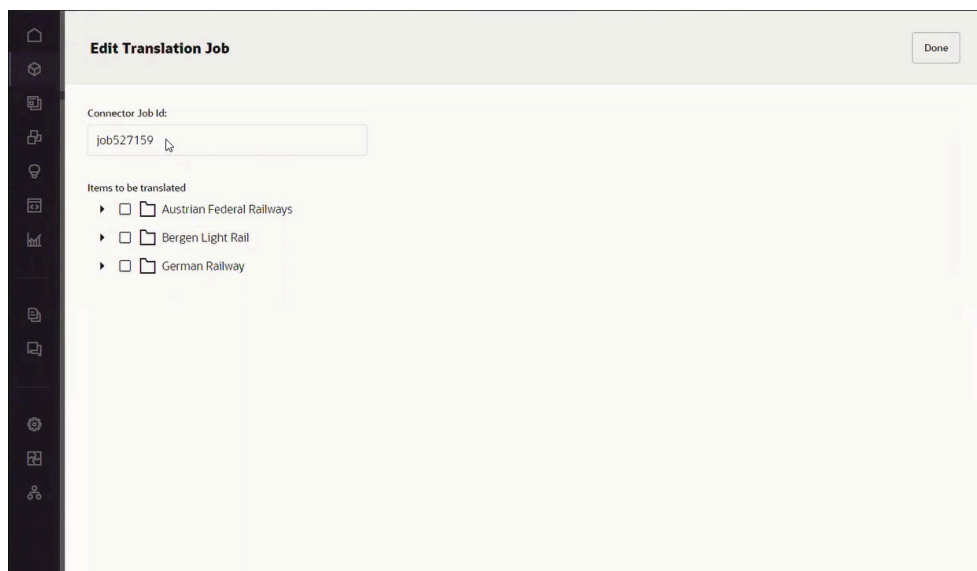
4. Access the **Content** page, select a repository and add the translation connector you've created.

Create a Translation Job and Render the Editor

1. On the **Assets** page, select an **Asset Type**. Under **Languages > Translate**, create a translation job. Enter a **Name**. Select a **Target Language** and **Translation Project**. Finish with **Create**.



2. The out-of-box Translation Job Editor user interface shows the basic information about your translation job. You can customize the editor in the JavaScript code.



Customize the Translation Job Editor

You can add more functionalities to the Translation Job Editor, customize it by using the Custom Translation UI SDK. Select your Translation Job Editor and find the out-of-the-box code in the `assets/js` folder. The provided default `main.js` is where `window.translationEditorSDK` is called. The file `util.js` is where a few helper functions are located.

Translation Jobs Validation

The custom translation connector, which is part of the Oracle Content Management (OCM) Toolkit, can return validation data of a translation job. The validation data indicates whether the documents in the translation job are translatable. The validation data is returned after the connector processes and uploads the documents to the Language Service Provider. The documents can include page content, fields or attributes of content items and digital assets, and native files of digital assets.

Behind the scenes, the custom connector implementation returns the validation data in the form of a JSON object through the `notificationData` property of the `TranslationJobInfo`. The same validation data is also presented in the OCM web UI on the translation jobs details page through the `GET_CONNECTOR_JOB_INFO` server API. Note that the Lingotek translation connector now also has the same feature.

The steps to enable this feature consists of the following:

- [Create and Start the Custom Translation Connector](#)
- [Configure the Custom Translation Connector](#)
- [Create a Translation Job to See the Validation Data](#)

Create and Start the Custom Translation Connector

1. Install the [OCM toolkit](#)

 **Note:**

The custom translation connector toolkit commands.

```
cec | grep translation-connector
```

<code>cec create-translation-connector <name></code>	Creates translation connector <name>. [alias: ctc]
<code>cec start-translation-connector <name></code>	Starts translation connector <name>. [alias: stc]

2. Create a translation connector
`cec create-translation-connector <name>`
3. Start a translation connector
`cec start-translation-connector <name>`

 **Note:**

Take note of your IP address of the desktop or Linux box running the toolkit. You will need it for later.

Configure the Custom Translation Connector

1. Configure the custom translation connector in the OCM web UI.

 **Note:**

The translation connector must be started before you can configure the custom translation connector in the OCM web UI.

To configure a custom translation connector, access the **Integration** page and select **Translation Connectors** in the top dropdown menu. Select **Create > Custom Connector**. Enter the required fields: **Name** {name of your translation connector}, and **Connector service URL** `http://{IP address:port}/connector/rest/api`.

Click **Next** to fill in the **Additional Fields** section as shown below.

- **Bear Token:** `Bearer token1`
- **WorkflowId:** `machine-workflow-id`
- **Additional Date:** `[]`

Once finished, **Save** and **Enable**.

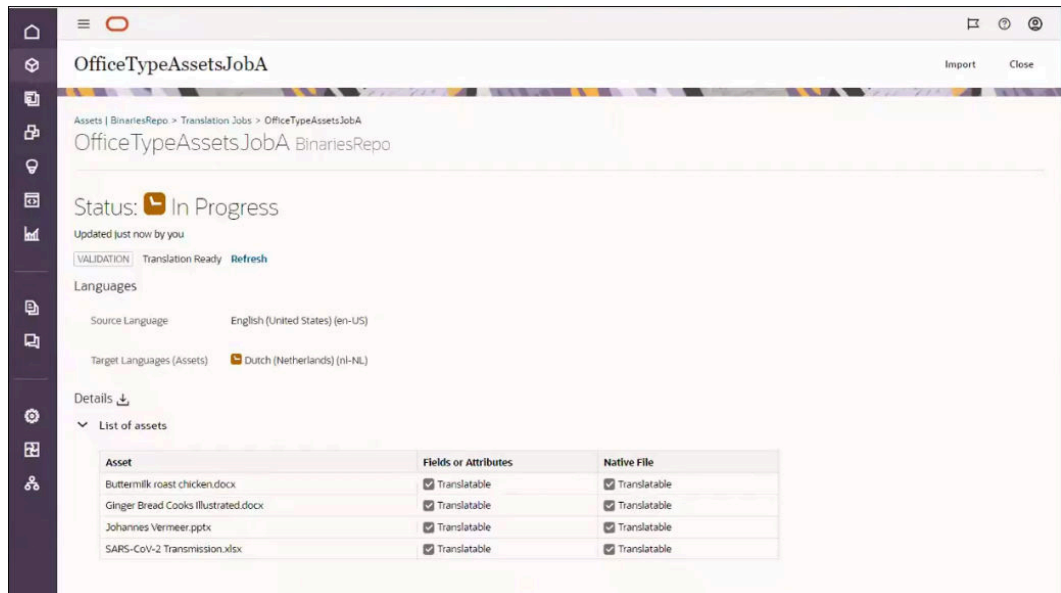
2. Use the custom translation connector by associating it with a repository.

Access the **Content** page, select a repository. Click **Edit** to add the translation connector you've created and configured.

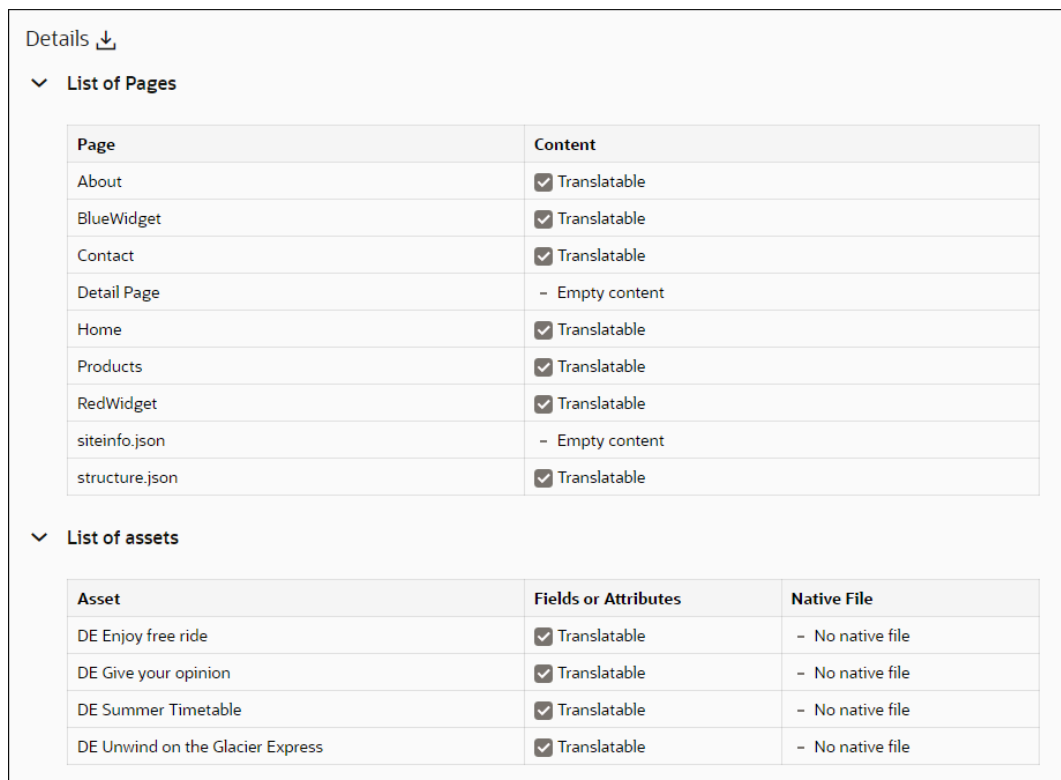
Create a Translation Job to See the Validation Data

1. On the **Assets** page, select an **Asset Type**. Under **Languages > Translate**, create a translation job. Enter a **Name**. Select a **Target Language** and **Translation Project**, which is the connector that's created in the previous steps. Finish with **Create**.
2. Go to the **Translation Jobs** page and select your translation job, expand the **Details** section to see the validation data.

This screen capture shows the page that lists the information about your particular translation job. In the Details section, you're able to see the validation data which indicates whether your documents are translatable.



This screen capture below shows the validation data retrieved by the translation connector.



Sample Translation Connector Implementation

You can use Content Toolkit to create the Sample Translation Connector, which provides a working connector that runs against a mock server.

You can use this sample implementation of a translation connector to help build your own custom translation connectors. The sample translation connector works with a mock Language Service Provider to simply add "##{locale}##" to each string to represent the "translation" of the string that an actual Language Service Provider would do.

The sample translation connector is built using the Connector Framework JS SDK provided by Oracle Content Management as part of Content Toolkit.

To install Content Toolkit, follow the directions listed on [GitHub](#).

The following sections describe how to develop a sample translation connector:

1. [Create the Sample Translation Connector with Content Toolkit](#)
2. [Register the Sample Translation Connector](#)
3. [Test the Sample Translation Connector](#)

Create the Sample Translation Connector with Content Toolkit

The `create-translation-connector` and `start-translation-connector` commands will create a named connector and install the required file to run the connector.

The connector contains a NodeJS `server.js` that you can use to run the Sample Translation Connector.

Register the Sample Translation Connector

Now that you've created the Sample Translation Connector, you can register it.

The registration of the connector can point to any Oracle Content Management server, such as the sample server:

```
> cec register-translation-connector connector1-auto -c connector1 -s
http://localhost:8084/connector/rest/api -u admin -p <password> -f
"BearerToken:Beartoken1,WorkflowId:machine-workflow-
id,AdditionalData:{}"
```

To register the translation connector through the Oracle Content Management Administration web interface:

1. Sign in to your Oracle Content Management instance as an administrator or developer.
2. Go to **Administration > Integration > Translation Connectors** and click **Create**.
3. Enter the details to register your sample translation content connector.
 - a. **Name:** Sample Connector – Machine Translation
 - b. **Description:** Connector to translate OCE assets and sites

- c. **Connector Service URL:** `http://host:port/sample-connector/rest/api` (a URL tested previously)
 4. Click **Next**.
 5. Click the **Additional Fields** tab. Fill in the fields and **Save** before exiting the page.
 - a. **BearerToken:** `Bearer token1`
 - b. **WorkflowId:** `machine-workflow-id`
 - c. **Translation Job Editor:** In the dropdown menu, find the name of Translation Job Editor that you've created and promoted.

 **Note:**

This field is optional and will appear only if you have configured custom translation UI. For details, see [REST Interfaces for Configuration and Authorization](#).

6. Check **Enable for end users** on the **General** page. On the **Integration** page > **Translation Connectors**, the **Enable** button is also available to enable your sample connector.

The default timeout values for the translation connector are set in the following two properties:

```
ConnectorConnectionTimeout=20000  
ConnectorReadTimeout=30000
```

If you want to change the values of these properties, you can add the properties to your `config.cfg` file and then modify either or both values.

Test the Sample Translation Connector

After you create and test a translation connector in Content Toolkit, you can register, enable, and test it in the Administration web interface.

 **Note:**

Before you can test the translation connector in the Oracle Content Management UI, associate the connector with a repository. In the left panel, go to **Content > Repositories**, and select your repository. In the dropdown of Translation Connectors, select the connector that you've created. **Save**

To test the translation connector in the OCM UI:

1. Sign in to Oracle Content Management as an Administrator or Developer.
2. In the navigation menu for the **Administration** web interface, click **Assets** on the top left, and then click an asset to select it.
3. Click **Translate** in the context menu.

4. In the create dialog, choose **< Sample Connector – Machine Translation >** in the dropdown menu.
5. Complete the create dialog.
6. Navigate to the **Translation Jobs** page.
When you view the translation job, it will now show you the % complete for the job.
7. Refresh the page until it shows 100% complete.
8. Import the translations from the connector.
Upon successful import, your asset will be translated into your selected language or languages.

Content Toolkit also provides a framework to test your Sample Translation Connector. The Sample Translation Connector itself comes with a sample Oracle Content Management translation job file under `data/translationBundle.zip`, which you can use for testing.

To run your Sample Translation Connector, bring up the Content Toolkit develop pages, select the connector, and then click through each of the steps:

```
> cec develop &
> open http://localhost:8085/public/translationconnections.html
```

Select the Sample Translation Connector you registered, and it will lead you to a page that lists the following steps to test the connector:

1. `Get Server Config` returns information about your connector that will be used to register the connector with Oracle Content Management.
2. `Create a job` creates a unique ID for the job. The sample connector creates an `out` folder with the same name as the job ID in the connector.
3. `Send zipfile to job` submits `data/translationBundle.zip` to the job for translation. The sample connector saves this file in the `out/{jobid}` folder, extracts the files, and then submits them for translation.
4. `Get the details of the job` asks for the status of the job.
5. `Get the translated zip` returns the created OCE Translation Job Translated Zip File, with the translation, once the job has moved to `TRANSLATED`.
6. `Delete the job` removes the `out/{jobid}` folder.

Understand the Sample Translation Connector Source Code

The source code of the sample translation connector contains REST APIs and a job manager to handle creation and persistence of the connector jobs.

The sample translation connector contains a set of REST APIs implemented using the Oracle Content Management connector interfaces. This is implemented as per the JAX-RS specification.

The job manager uses `persistenceApi` to store the following resources:

- Metadata about the job in the connector and the project created in the Language Service Provider
- Contents of the translation zip file from Oracle Content Management

- Metadata about each file and the corresponding document in the Language Service Provider
- Combination of the translations with the original files
- Creation of the translated zip file for the connector

The `persistenceApi` included in the sample simply uses the files system to manage these resources. A full translation connector implementation will need to make sure that the persistence layer will continue to work during failover and upgrade as well as provide secure access to the resources stored.

The job manager also maintains information in the connector job metadata file about the status of a translation job in the Language Service Provider by querying the information back either via polling or via a callback.

Code Structure:

connector:

- *SampleConnectorRouter.js*
 - Authenticates the request
 - Extracts parameters from the request URL
 - Maps the request URL to the corresponding `SampleConnector` function:
 - GET: `/api/connector/v1/server` => `getServer()` - metadata about the connector
 - POST: `/api/connector/v1/job` => `createJob()`
 - GET: `/api/connector/v1/job/:id` => `getJob()`
 - POST: `/api/connector/v1/job/:id/translate` => `translateJob()` - accepts a zipfile
 - GET: `/api/connector/v1/job/:id/translation` => `getTranslation` - get translated zip
 - DELETE: `/api/connector/v1/job/:id` => `deleteJob()`
- *SampleConnector.js*
 - Validates the request parameters
 - Calls the appropriate connector code
 - Formats the response

- *SampleBasicAuth.js*

connector/job-manager:

- *sampleJobManager.js*
 - Expand the zip file and then, for each file:
 - * Filter the file to include only translatable fields
 - * Submit the translatable fields to the LSP
 - * Wait for the translations to come back (either by polling or by callbacks)
 - * Recombine the translatable fields with the original files
 - * Create a zip file of all the translations
 - Also restarts any running jobs on startup to support failover

- *sampleFileImporter.js*
 - Imports the filtered document into the LSP specifying the source language
- *sampleFileTranslator.js*
 - Asks the LSP to translate the documents into the target languages
- *sampleFileDownloader.js*
 - Downloads translations for a document

connector/apis:

- *persistenceApi.js*
 - Basic implementation using the filesystem to:
 - * Unpack the zip file and maintaining the source
 - * Accept translated files and creates the expected zip structure
 - * Zip up and return the translated files
- *filterApi.js*
 - Extracts translatable strings from the source files based on file type
 - Recombines the translated strings with the source file
- *connectorApi.js*
 - Constructs the requests to the LSP
- *httpApi.js*
 - Makes the actual GET/POST requests to the LSP

MockServer:

- *mockLSPServer.js*
- *mockBearerAuth.js*
 - A mock LSP server
 - It expects the following headers:
 - * BearerToken – can be any `Bearer #####` value
 - * WorkflowId – supports only `machine-workflow-id`

Translation Job Original Zip File Format

The connector is responsible for extracting the files to be translated from the translation zip file, filtering them, and sending them to the LSP. To do this, the connector needs to understand the format of the zip file.

There are two types of translation jobs in Oracle Content Management, so there are two file structures possible in the zip file:

- Asset Translation
 - Zip file format
 - * "job.json" - Contains details on the source & target languages.
 - * "root"

- * <asset files to be translated>
- Site Translation
 - Zip file format
 - * "site"
 - * "job.json" - Contains details on the source and target languages.
 - * "root"
 - * <site files to be translated>
 - * "assets"
 - * "job.json" - Contains details on the source and target languages.
 - * "root"
 - * <asset files in the site to be translated>

Translation Job Translated Zip File Format

On successful translation, the connector must recombine the translations with the original files and create a zip file in the following format, which is based on the original format.

- Asset Translation
 - Zip file format:
 - * "job.json"
 - * "de-DE"
 - * <asset files>
 - * "fr-FR"
 - * <asset files>
 - * "it-IT"
 - * <asset files>
 - * "root" (Original files can optionally be included. The directories also need to be included, and not just the files.)
 - * <asset files>
- Site Translation
 - Zip file format:
 - * "site"
 - * "job.json"
 - * "de-DE"
 - * <site files>
 - * "fr-FR"
 - * <site files>
 - * "it-IT"
 - * <site files>

- * "root" (Original files can optionally be included. The directories also need to be included, and not just the files.)
 - * <site files>
- * "assets"
 - * "job.json"
 - * "de-DE"
 - * <asset files>
 - * "fr-FR"
 - * <asset files>
 - * "it-IT"
 - * <asset files>
- * "root" (Original files can optionally be included. The directories also need to be included, and not just the files.)
 - * <asset files>

Develop External Processors

The Content Capture feature of Oracle Content Management provides the ability to integrate custom functionality. This custom functionality is referred to as an external processor and is hosted separately from Content Capture.

To connect an external processor to Content Capture, one adds an External Processor Job step in a procedure flow. For details, see *Configure External Processor Jobs* in the *Capturing Content with Oracle Content Management* guide. When a batch of documents reaches this step, it is locked to that specific step in the flow. The external processor is then responsible for processing those documents. After the processing of documents is completed, the external processor must inform Content Capture of the processing results for each document, regardless of whether it was completed successfully or failed. If the external processor fails to notify Content Capture it has finished processing a document, Content Capture will eventually time out the processing and fail the batch.

An external processor could do various processing on documents. Perhaps there is an existing application that scans a document for signature blocks to parse out company information. An external processor could wrap that capability enabling it to be called from a procedure flow. Metadata values extracted by the application could then be set into the corresponding Content Capture document's metadata fields. Or maybe there is an onsite database that contains discount codes for various vendors. An external processor could look up codes by a Content Capture metadata field of the document and set the value in a different metadata field. An external processor could be developed to perform virtually any type of desired processing on a Content Capture document.

The following sections describe how to develop an external processor:

- [External Processor Execution Flow](#)
- [REST API for Content Capture](#)
- [External Processor SDK](#)
- [External Processor Examples](#)

External Processor Execution Flow

An external processor can operate in two modes of execution: push versus pull.

In the push mode, Content Capture pushes documents to the external processor. In this mode, the external processor responds to API events that come to it. Conversely, in pull mode, the external processor pulls Content Capture to find new work.

The push model of operation requires the external processor to implement a REST endpoint service definition that Content Capture expects. The push model introduces additional networking complexities since the receiving external processor must be publicly accessible over the Internet. Content Capture is restricted to only sending outbound requests via HTTPS thus requiring external processors implementing the push model to receive requests over HTTPS.

An external processor implemented in the pull model does not have these additional networking challenges. It can easily be placed behind a corporate firewall while making its

calls outbound to the publicly accessible Content Capture APIs. The pull model imposes the additional work on the external processor of polling for new work.

The push and pull models are explained in the following sections:

- Push Model

 **Note:**

The push model is deprecated and under maintenance. The documentation will be available when the push model is updated and re-released.

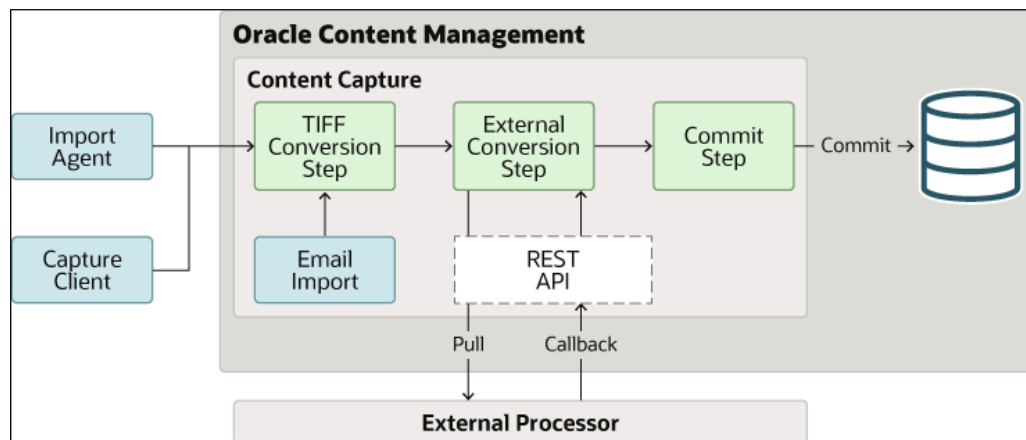
- Pull Model

Pull Model

In the pull model, the external processor has the responsibility of polling for documents to process which it does by using the standard Content Capture API.

Since the processing is controlled by the external processor, the pull model is always considered asynchronous processing. Similarly, the external processor can make multiple Content Capture REST API calls to manipulate the state of the document in Content Capture. Once all the changes are finished, the external processor releases the lock on the document with the complete API call.

The figure below shows the general flow of processing:



REST API for Content Capture

You can use the Oracle Cloud REST API for Content Capture for the manipulation of documents when they are locked to an external processor job step. This includes changing a document's metadata field values, adding or removing attachments to documents, and even deleting documents.

The [REST API for Content Capture](#) has several categories of endpoints, which are described in the following table.

Category	Description
Attachments	Manages document attachments in Capture
Auditing	Queries and analyzes batch and document processing
Batches	Manages batches in Content Capture
Documents	Manages documents in Capture
Steps	Queries and updates processing task queues
System	Provides system-level operations

External Processor SDK

To help build a new external processor, Oracle Content Management provides a Content Capture SDK. The SDK contains:

- The Javadoc of the SDK.
- A JAR file containing the SDK.
- A JAR file containing the Javadoc, useful with Java IDEs.
- A JAR file containing the SDK source, useful with Java IDEs.
- The `capture-rest-api-v1.yaml` file, which documents the REST API for Content Capture.
- The `external-processor-rest-api-v1.yaml` file, which documents the REST API for external processors.

The Content Capture SDK is in the file `capture-sdk.zip` or `capture-sdk.tgz`. You can download the file from here: <https://your-ocm-service/capture/api/sdk/capture-sdk.tgz> or <https://your-ocm-service/capture/api/sdk/capture-sdk.zip>.

For more details, see [REST API for Content Capture](#).

The SDK is only provided in Java. It requires JAX-RS 2.1 for the RESTful services, [Eclipse Jersey 2.31](#) for the implementation, and the JSON library [Jackson 2.12.3](#) for mapping between JSON and Java classes.

External Processor Examples

Three examples of the external processors can be downloaded. The examples file contains:

- The Java source code of the external processors.
- The `External-Processor-Examples.xml` file, which provides an exported sample Content Capture Procedure.

The examples are in the file `capture-examples.zip` or `capture-examples.tgz`. You can download the file from here: <https://your-ocm-service/capture/api/sdk/capture-examples.tgz> or <https://your-ocm-service/capture/api/sdk/capture-examples.zip>.

The examples illustrate three different models of interaction between Content Capture and an external processor. The first two examples are of the push model and illustrate both the synchronous and asynchronous response types. The third is an example of the pull model.

 **Note:**

Only Example 3 is described in this section as seen below. The first two examples use the push model, which is deprecated and under maintenance. The documentation for Examples 1 and 2 will become available when the push model is updated and re-released.

The included Content Capture Procedure provides sample configurations for calling the examples. It contains the necessary metadata fields, document profiles, and job definitions. It also provides a Capture client profile to create and release documents into the Procedure flow.

After the Content Capture Procedure is imported:

- In the Example 3 code, the unique Processor Step identifier will be updated by the import and its new value will need to specify on the command line as a standard Java property (i.e. `-Doracle.ocm.capture.examples.externalprocessor.step-id=54cd7e16-5440-46a7-a5d4-6878b38c8b97`).

Example 3: Resize Image (Pull Model)

This example shows how to develop an external processor using the pull model. This external processor will accept an image document which it will scale by a given percentage and then return the scaled document to replace the original document in Content Capture. The following elements are illustrated:

- The external processor periodically polling Content Capture for documents that need processing.
- When the documents are available, the Content Capture responds with a `document` JSON object containing the metadata fields associated with each waiting document, including a scale value.
- For each waiting document, the external processor:
 - retrieves the image file using the Capture `get document content` API method
 - resizes the image document by the scale value
 - sends the resized image file back to Content Capture to replace the original file using the `document update` API method
 - indicates the processing of the document is completed using the Capture `complete document task` API method

There is one metadata field used for the example is `Scale`. It needs to contain a value between 1 to 100 that represents the percentage to scale or resize the image from its present size.

11

Compile Content Layouts as HTML

In some cases, you may want to use an asset as its HTML rendition in associated content layout in web applications or other discrete ways, without precompiling an entire site. To publish a layout as HTML:

- Enable **Publish HTML** when creating a content or asset type in the Oracle Content Management web interface. For information on how to create a content or asset type, see *Create a Content Type in Managing Assets with Oracle Content Management*.
- Select a component layout that can compile the layout as HTML, meaning it must have a `compile.js` script in the assets directory. For information on content layout compilers, including sample code of a content layout compiler, see *Content Layout Compilers in Building Sites with Oracle Content Management*.
- Publish the asset to trigger compilation of the layout. When an asset of that type is created and published, the HTML renditions are generated and made available in the REST API at:

```
<Instance_URL>/content/published/api/v1.1/items/<item_guid>/renditions?  
channelToken=<token>
```

- You can access the HTML rendition log. The log is generated after assets that are configured to generate HTML renditions are submitted for publishing. The rendition log can be downloaded from **Assets** in the Oracle Content Management web interface, then go to **Publishing Events Logs**. Logs are JSON files that contain either detailed information on the generated renditions and the layout used for the renditions, or error messages that can help you troubleshoot issues if compilation failed for some reason.

Note:

You can compile content layouts only if your Oracle Content Management instance is running natively on Gen 2 Oracle Cloud Infrastructure (OCI).

Part IV

Developing Oracle Content Management Integrations

This part provides information on developing Oracle Content Management integrations. It includes the following chapters:

- [Understand Cross-Origin Resource Sharing \(CORS\)](#)
- [Embed the Web User Interface in Other Applications](#)
- [Oracle Content Management REST APIs](#)
- [Oracle Content Management SDKs](#)
- [GraphQL Support in Oracle Content Management](#)
- [Use Webhooks](#)
- [Set Proxies](#)

12

Understand Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing (CORS) allows a web page to make requests such as XMLHttpRequest to another domain. If you have a browser application that integrates with Oracle Content Management Cloud but is hosted in a different domain, add the browser application domain to Oracle Content Management Cloud's CORS origins list.

The REST APIs use CORS because they're called from JavaScript code that runs in a browser and the REST APIs and Oracle Content Management are hosted in different domains.

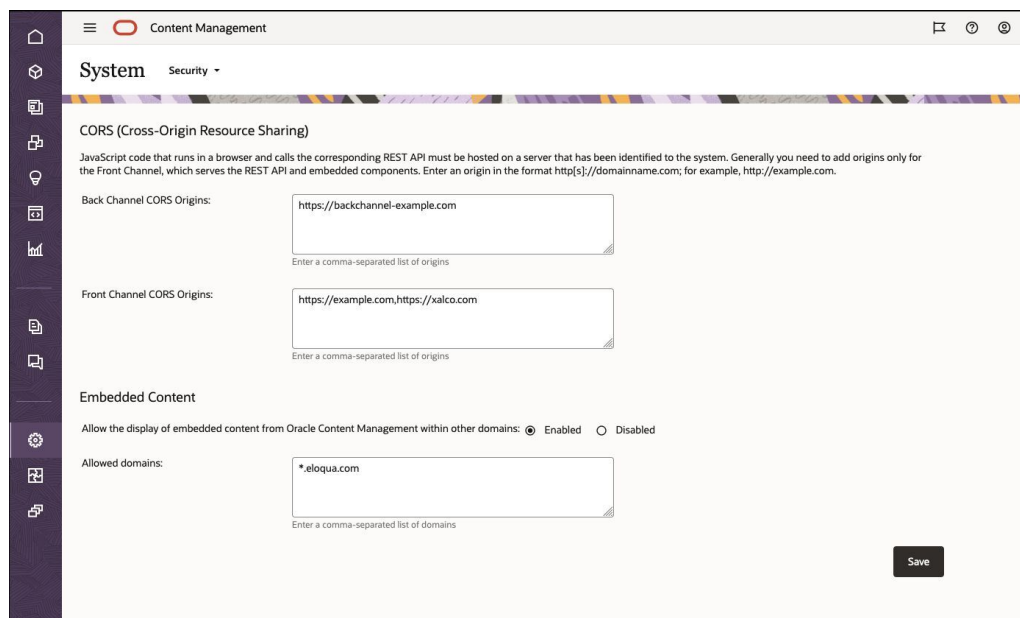
If your browser application needs to use a REST endpoint that doesn't support CORS or that needs service account credentials, you can instead register and use the endpoint via Oracle Content Management's integrated proxy service. See [Configure Proxy Service Settings](#).

In general, inline frames can host content if the protocol, domain, and port of the inline frame are identical to those for the content it displays. For example, by default, an inline frame on the page `http://www.example.com:12345/home.html` can host content only if the content's protocol is also `http`, the domain is `www.example.com` and the port is `12345`.

However, if the application is in a different domain than Oracle Content Management, you need to add the application's host machine information to the list of front-channel CORS origins, back-channel CORS origins, or both.

- If the request is a cross-domain request (not originating from Oracle Content Management's domain) that will be served by Oracle Content Management, you need to add a front-channel CORS origin. Front-channel CORS is typically useful for custom application integration. For example, the REST APIs interact with the front channel.
- If the request is directly from Oracle Content Management to a connected client in another domain, you need to add a back-channel CORS origin. For example, Oracle Content Management can send back-channel messages (real-time updates) to an application.
- If an application gets both front-channel and back-channel communication from Oracle Content Management, you need to add the domain to both the front and back channel CORS origins lists.

The CORS settings apply to all Oracle Content Management calls (documents, social, and content as a service).



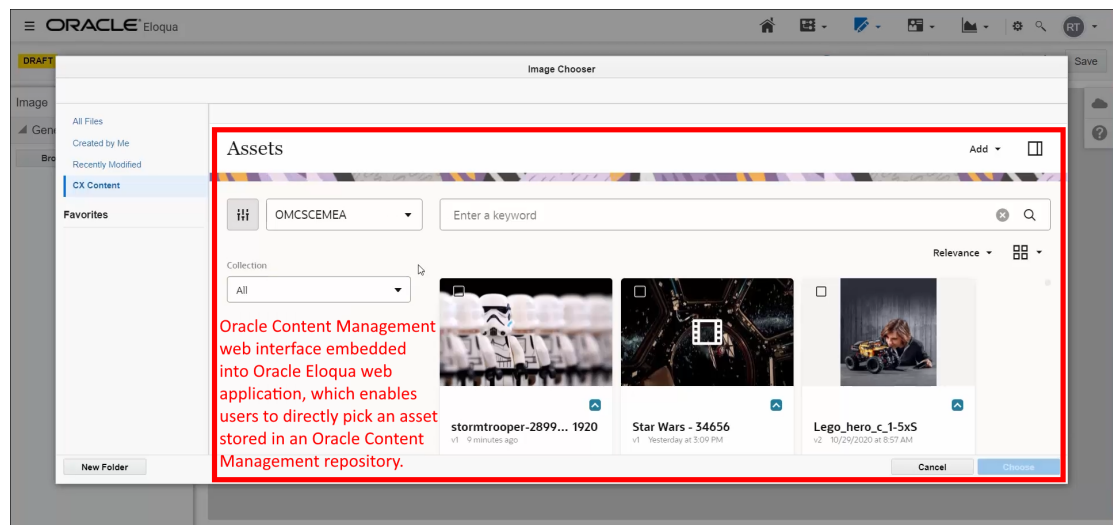
See [Enable Cross-Origin Resource Sharing \(CORS\)](#) in *Administering Oracle Content Management*.

13

Embed the Web User Interface in Other Applications

The [Embed UI API V2 for Oracle Content Management](#) is a JavaScript API that enables you to embed the Oracle Content Management web user interface into your own web applications using an HTML inline frame (iframe tag). The JavaScript API simplifies the creation of the iframe element and manages communication with the code running in the frame. The embedded web interface can include asset and document lists, conversations, site content, search results, and other Oracle Content Management features.

Here's an example of what the embedded UI could look like (an Oracle Content Management asset selection list inside the Oracle Eloqua web application):



You configure the web user interface by passing a JavaScript options object into the API. In addition, you can provide callback functions to handle various events that occur when a user interacts with the embedded web user interface.



Note:

The current Embed UI API version is V2, which you should use for all new implementations. V1 has been deprecated, but it will remain [available](#) until further notice. Sometime in the future, V1 will no longer be supported.

Browser settings that affect third-party cookies can impact embedding of the Oracle Content Management user interface, including single sign-on (SSO). Users may need to adjust their web browser preferences to enable cross-site tracking for the embedded UI to work correctly.

The Embed UI API V2 supports embeddable components to control these user interface elements:

- [Assets view](#)
- [Asset viewer](#)
- [Content item editor](#)
- [Content item create](#)
- [Documents view](#)
- [Document viewer](#)
- [Conversations list](#)
- [Conversation view](#)

In addition, you can set a number of [general options](#) and also customize [dialogs](#) in the embedded web user interface.

See [Embed UI API V2 for Oracle Content Management](#) to learn more including tutorials and reference materials.

Oracle Content Management REST APIs

REST APIs are available in Oracle Content Management for content delivery and for management of content, conversations, documents, and users and groups.

Oracle Content Management has several REST APIs, listed in the following table.

API Name	Description
REST API for Activity Log	Provides the ability to search activities in Oracle Content Management.
REST API for Content Delivery	Provides access to published assets in Oracle Content Management. Published assets include content items and digital assets, as well as their renditions.
REST API for Content Capture	Provides several operations for the manipulation of documents when they are locked to an external processor job step. This includes changing a document's metadata field values, adding or removing attachments to documents, and even deleting documents.
REST API for Content Management	Provides access to manage assets in Oracle Content Management. Assets include content items and digital assets and their renditions.
REST API for Content Preview	Provides the ability to preview items, item variations, items by slug, renditions, and renditions by slug.
REST API for Conversations	Enables interaction with your cloud resources for real-time collaboration between individuals and teams to connect your business processes, enterprise applications, and content.
REST API for Documents	Enables you to interact with folders and files stored in Oracle Content Management and to create sites from templates and other sites.
REST API for Self-Management	Provides the ability to view the authenticated user details.

API Name	Description
REST API for Sites Management	<p>Provides the ability to create sites from templates and then manage the life cycles of those sites in Oracle Content Management. This REST API has several categories of endpoints:</p> <ul style="list-style-type: none"> • Components Import, export, list, read, copy, update, publish, share, and delete components. • Themes Import, list, read, copy, publish, update, share, and delete themes. • Templates Create policies and associate them with a template to be used when creating a site. Policies can be used to enforce site creation approval, who can use a template, and what security the site will have. • Policies Manage policies, and manage the membership of policies with access lists. • Requests Requests are made when creating a site. A site request is rejected or approved through a review. • Sites Create, copy, delete, publish, and activate sites. Get site information and associated resources. List sites, and manage membership.
REST API for Users and Groups	Enables you to manage users and groups of users.
REST API for Webhooks Management	Provides the ability to manage webhooks in Oracle Content Management.

The following topics provide descriptions of how to perform some tasks with REST APIs:

- [Integrate with Oracle Content Management Using OAuth](#)
- [Download the Swagger File for a REST API](#)
- [Upload a REST API Swagger File into Mobile Cloud Service](#)
- [Search with the Querytext Parameter](#)
- [Set Up Searches on Metadata Fields](#)
- [Create and Use Applinks for File and Folder Access](#)
- [Provide Access to Files and Folders with Public Links](#)

Integrate with Oracle Content Management Using OAuth

Developer access to Oracle Content Management (OCM) REST APIs for administering system or managing assets, files, or other resources is supported only via token-based authorization. To access the OCM REST APIs, you need an OAuth 2 access token to use for authorization.

You need to create an OAuth client application to be able to request the access token. OAuth client is simply an HTTP client that must be registered as an OAuth 2 client using the Oracle Identity Cloud Service (IDCS) or Identity and Access Management (IAM) with Identity Domains administration console.

If single sign-on (SSO) is already established with IDCS (the same that OCM points to), then we do not need OAuth. The SSO should be sufficient since OCM REST APIs support SSO as well. If the UI will not have an active SSO session, then OAuth should be used.

Applications can request an access token to access OCM REST APIs in different ways depending on the grant type specified in the client application. A grant is a credential representing the resource owner's authorization to access a protected resource. The OCM REST APIs have been certified to work with the following three grant types:

- **Client Credentials:** When using the Client Credentials grant type, the client authenticates with the OAuth service and then requests an access token. With this grant type, the OAuth client application presents its own credentials to obtain an access token for the client-initiated requests to access a protected resource. This access token is either associated with the client's own resources, and not a particular resource owner, or is associated with a resource owner for whom the client is otherwise authorized to act. Trusted applications (such as back-end services) may request access tokens directly on behalf of users. This is an OAuth two-legged authorization flow. Another use case of this flow or grant type is when integrating application users are not users of OCM, (i.e. they reside in a different identity store other than OCM service's identity store and hence OAuth application client credentials are used to make API calls to OCM).

To know more about Client Credentials grant type, see [Client Credentials Grant Type](#).

- **Authorization Code:** The OAuth web applications typically need to first validate the user's identity and optionally obtain the user's consent. This is an OAuth three-legged authorization flow and in this case, the Authorization Code grant type is used. In this case, you want to obtain an authorization code first by using an authorization server as an intermediary between the client application and the resource owner. An authorization code is returned to the client through a browser redirect after the resource owner gives consent to the authorization server. The client then exchanges the authorization code for an access (and often a refresh) token. The resource owner credentials are never exposed to the client. To know more about Authorization Code grant type, see [Authorization Code Grant Type](#).
- **Resource Owner:** When using the Resource Owner grant type, the resource owner's password credentials (user name and password) can be used directly as an authorization grant to obtain an access token. Therefore, this grant type is used when the resource owner has a trust relationship with the client, such as a computer operating system or a highly privileged application, because the client must discard the password after obtaining the access token. To know more about Resource Owner grant type, see [Resource Owner Password Credentials Grant Type](#).

There are four steps that you must perform to use an OAuth client to access OCM REST APIs:

1. Log in to the Oracle Identity Cloud Service (IDCS) or IAM with Identity Domains administration console.
2. Create an OAuth client application and make note of the client ID and client secret.
3. Use the client ID and client secret to request an access token from the Oracle Identity Cloud Service OAuth Service.

4. Include the access token in the appropriate HTTP header when you make REST API calls.

For content management and sites management REST APIs, you acquire the client ID and secret required to get the access token from Oracle Identity Cloud Service (IDCS) or IAM with Identity Domains. For the two-legged OAuth, you use client credentials to acquire the client ID and secret. For the three-legged OAuth, you use an authorization code to acquire the client ID and secret.



Note:

The Oracle Cloud Console is merging IDCS into Identity and Access Management (IAM) with Identity Domains. You can check if your cloud account is using identity domain or not. Sign in to the Oracle Cloud Console. In the navigation menu, click **Identity & Security**. Under **Identity**, check for **Domains**. If you see Domains, then your cloud account has been updated. For details, see [Oracle Cloud Infrastructure Documentation](#).

The following topics describe how to use your account:

- [Cloud Account Using IAM Identity Domain](#)
- [Cloud Account Using Oracle Identity Cloud Service](#)

Cloud Account Using IAM Identity Domain

The following sections describe how to use your cloud account:

- [Access OCM Using Client Credentials \(Two-Legged OAuth in Identity Domain\)](#)
- [Access OCM Using Authorization Code \(Three-Legged OAuth Flow in Identity Domain\)](#)
- [Access OCM Using Resource Owner \(Identity Domain\)](#)

Access OCM Using Client Credentials (Two-Legged OAuth in Identity Domain)

For this OAuth flow, the token is requested for the client application and calls to Oracle Content Management (OCM) APIs are made using this client application's token. This OAuth client application needs to be added to the OCM roles to have appropriate OCM access.

The **Identity Domain Administrator** or **Application Administrator** of the tenant creates an OAuth client for the developer with the required privileges.

The **Identity Domain Administrator** performs the following steps to get an access token:

1. [Create an OAuth Client and Acquire a Client ID and Secret](#)
2. [Grant the Required Oracle Content Management Roles to the Client](#)
3. [Acquire an Access Token from Identity Domain for the Required Resource](#)
4. [Use the Access Token to Access the Oracle Content Management Resource](#)

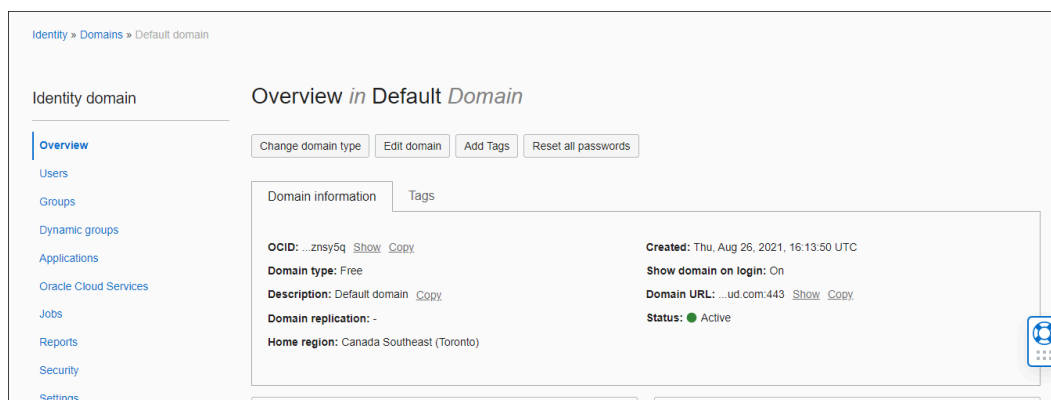
Create an OAuth Client and Acquire a Client ID and Secret

The Identity Domain Administrator or an Application Administrator of the tenant creates an OAuth client for the developer with the required privileges. To do this, perform the following steps:

1. Sign in to the Oracle Cloud Console.

`https://cloud.oracle.com`

Navigate to **Identity & Security**. Under **Identity**, check for **Domains**. Select the domain (default or newly created) while creating an application, you will see the **Applications** option in the left menu as shown in the figure below.



2. Click to add a new application in the **Applications** section.
3. Select **Confidential Application**. The Confidential Applications use OAuth 2.0 and can protect their OAuth client ID and client secret. To know more about Confidential Applications, see [Add a Confidential Application](#).
4. Give the application a name and, optionally, a description, and click **Next**.
5. Choose to configure this application as a client.
6. Select the **client credentials** grant type on the **Authorization** screen.
7. Scroll down to the **Token Issuance Policy** section and, under **Resources**, click **add scope** to give the application access to the required Oracle Content Management instance.
8. Click the right arrow to select the scope.
The only scope required is the one ending in `urn:opc:cec:all`.
Select the checkbox next to it and then click **Add**.
9. Click **Next** until the end of the train.
The **Resources**, **Web Tier Policy**, and **Authorization** steps are related to applications that have some resource authenticated or authorized by Identity Domain; for example, a web application. This isn't relevant in the case of a simple server-server client, which is discussed here.
10. Click **Finish**.
11. Note the **Client ID** and **Client Secret** values because you'll need those to get a token later.

12. Check **Activate** and then click **Save** to enable the application.

You can decide how to handle the client ID and secret.

 **Note:**

Just as a regular user, an OAuth client with the client credentials requires relevant permission on objects in your Oracle Content Management instance to be able to perform actions using REST API for Content Management. For example, it needs to be a repository member with Contributor permission to create assets in it, or a Manager on the repository to modify its settings. In the Oracle Content Management web UI, you can use the Members dialog to find your OAuth client by name and add it to the required object. A newly created OAuth client will appear in search only after the next hourly Identity Domain sync is complete.

Grant the Required Oracle Content Management Roles to the Client

You can grant the required roles through the Oracle Cloud administration console.

1. Sign in to the Oracle Cloud administration console.
2. Click **Oracle Cloud Services**.
3. Select the Oracle Content Management service instance for which you need privileges. This instance name would have been provided by your account administrator.
4. Click **Application Roles** and then right-click the burger menu on the right.
5. Choose **Assign Applications**.
6. Assign the client application to the roles based on the access that is required to OCM objects. For OCM application roles, see [Application Roles](#).
7. Save the changes.

Acquire an Access Token from Identity Domain for the Required Resource

Use the client ID and secret obtained earlier from the OAuth client to get an `AccessToken` for the resource.

A sample curl command to acquire an access token follows:

AccessToken Request

```
curl -X POST \
https://<idcs tenantname that is protecting this service>/oauth2/v1/
token \
-H 'Authorization: Basic <Base64 encoded clientID:clientSecret>' \
-d 'grant_type=client_credentials&scope=<scope string>'
```

For scope string, see Step 7 of [Create an OAuth Client and Acquire a Client ID and Secret](#). For Example:

```
curl -X POST 'https://
idcs-99463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/
```

```

oauth2/v1/token' \
  -H 'Authorization: Basic
Y2RlNGQyMDZjYTE1NGM2Yjg2NGMyMTJmMjY3MmE6MWM0ZTAyNGQ0tZDU4MC00MzEzLWJkZjMtZ
TQyMGQzMzgZTY2' \
  -d'grant_type=client_credentials&scope=https://
50B904E53A6F4EA1B45F30E225EA99B.cec.ocp.oc-test.com:443/urn:opc:cec:all'

```

Identity Domain returns the key and value for an access token:

```
{"access_token": "<access-token-value>"}
```

You can copy the access token value.



Note:

The refresh token is not included in the Client Credentials grant type flow.

Use the Access Token to Access the Oracle Content Management Resource

Once the access token is acquired, call the Oracle Content Management endpoint to access the resource.

A sample curl command follows.

```

curl -X GET <Oracle-Content-Management-URL>/sites/management/api/v1/sites \
-H 'Authorization: Bearer <Access Token acquired from previous step>' \
-H 'Content-Type: application/json'

```

Access OCM Using Authorization Code (Three-Legged OAuth Flow in Identity Domain)

In this flow, a client application is created to acquire the client ID and secret, which are used by the client browser. The client browser then redirects the user to Identity Domain to log in with the username and password. Upon successful authentication, an authorization code is sent with the redirect URL. An access token is requested from Identity Domain using this authorization code. Thus, the resource owner credentials are never exposed to the client in this flow.

Because this is a user token, the user requesting the token needs to have relevant Oracle Content Management roles as the user's access to the OCM objects will be checked against assigned roles.

The **Identity Domain Administrator** performs the following steps to get an access token:

1. [Create an OAuth Client and Acquire a Client ID and Secret](#)
2. [Grant the Required Oracle Content Management Roles to the Client](#)
3. [Acquire an Access Token from Identity Domain for the Required Resource](#)
4. [Use the Access Token to Access the Oracle Content Management Resource](#)

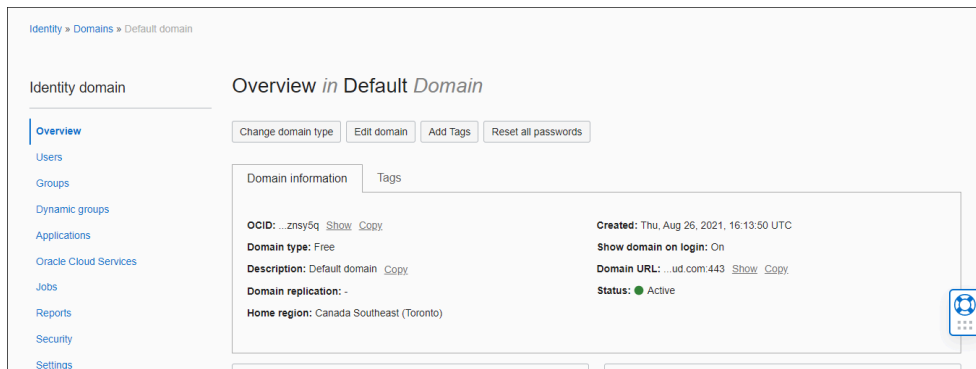
Create an OAuth Client and Acquire a Client ID and Secret

The Identity Domain Administrator or an Application Administrator of the tenant creates an OAuth client for the developer with the required privileges. To do this, perform the following steps:

1. Sign in to the Oracle Cloud Console.

`https://cloud.oracle.com`

Navigate to **Identity & Security**. Under **Identity**, check for **Domains**. Select the domain (default or newly created) while creating an application, you will see the **Applications** option in the left menu as shown in the figure below.



2. Click to add a new application in the **Applications** section.
3. Select **Confidential Application**. The Confidential Applications use OAuth 2.0 and can protect their OAuth client ID and client secret. To know more about Confidential Applications, see [Add a Confidential Application](#).
4. Give the application a name and, optionally, a description, and click **Next**.
5. Choose to configure this application as a client.
6. Select the **Authorization code** grant type on the **Authorization** screen.
7. Enter a **redirect URL** value to point to the URL where the user needs to be redirected after authorization.
8. Enter a **Post logout URL** value to point to the Oracle Content Management service administration console.

Note:

It is not a mandatory field while creating a client application with the Authorization Code grant type. As mentioned in the IDCS documentation (see [Using OpenID Connect for Log Out](#)), this could be again any client-specific URL to where the redirection happens after logout.

9. Scroll down to the **Token Issuance Policy** section and, under **Resources**, click **add scope** to give the application access to the required Oracle Content Management instance.
10. Click the right arrow to select the scope.

The only scope required is the one ending in `urn:opc:cec:all`.

Select the checkbox next to it and then click **Add** (note down the scope string as it will be used later on while requesting the access token).

11. Click **Next** until the end of the train.
The **Resources**, **Web Tier Policy**, and **Authorization** stops are related to applications that have some resource authenticated or authorized by Identity Domain; for example, a web application. This isn't relevant in the case of a simple server-server client, which is discussed here.
12. Click **Finish**.
13. Note the **Client ID** and **Client Secret** values because you'll need those to get a token later.
14. Check **Activate** and then click **Save** to enable the application.

The client can store the **Client ID** and **Client Secret** values securely in a credential store.

Grant the Required Oracle Content Management Roles to the Client

You can grant the required OCM roles to the user through the Oracle Cloud administration console.

1. Sign in to the Oracle Cloud administration console.
2. Click **Oracle Cloud Services**.
3. Select the Oracle Content Management service instance for which you need privileges. This instance name would have been provided by your account administrator.
4. Click **Application Roles**.
5. Right-click the burger menu on the right. Choose **Assign Users**.
6. Assign the user to the roles based on the access that is required to OCM objects. For information about OCM application roles, see [Application Roles](#).
7. Save the changes.

Acquire an Access Token from Identity Domain for the Required Resource

Acquisition of the access token consists of multiple steps:

1. The UI of the customer's application makes authorization code requests for the user to Identity Domain.
 - a. The UI makes a GET call to Identity Domain with the URL given below. No authorization is required (this needs to be accessed via browser as this will open login UI, Postman will not work in this case).

```
https://<IDCS-service-instance>/oauth2/v1/authorize?  
client_id=<clientid>&response_type=code&redirect_uri=<client-redirect-  
uri>&scope=<scope string from Step 9 of Create oauth  
client>&nonce=<nonce-value>&state=1234
```

Where `nonce-value` and `state` are optional.

Example:

```
https://
idcs-4c463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/
oauth2/v1/authorize?
client_id=51d06799e2aa4c749a79276cf7d24ca7&response_type=code&red
irect_uri=https://p2qaextradplsrvcl-qucm2019927ac99.cec.ocp.oc-
test.com/documents&scope=https://
50B904E53A6F4EA1B45F30E225E6C99B.cec.ocp.oc-test.com:443/
urn:opc:cec:all&nonce=12345&state=12345
```

- b. Identity Domain shows the login UI.
- c. The user logs in to the login UI. The Identity Domain redirects to redirect URL with the Authorization code as shown below:

```
<redirection_url>?code=<auth_code>&state=12345
```

Example:

```
https://p2qaextradplsrvcl-qucm2019927ac99.cec.ocp.oc-test.com/
documents?
code=AgAgZTM5MGMzODVjNzY3NGZhNzgzMmExMTI0YmU1ZDBiOTYIABC_fyt25KEc
HP9-Hmjkh9y1AAAAMPia8fMCbuzkdk10_CzsrqZ-
EtGwsI_MCPFr1Y3gt25EhrhYH65mb7vkJmPvUeKdTg&state=12345
```

2. The UI makes a call to Identity Domain to get the user token, passing `auth_code` obtained in the previous step with the following payload:

```
Authorization: Basic <Base64 encoded clientID:clientSecret>
Method: Post
URL: https://<IDCS-service-instance>/oauth2/v1/token
BODY (as form data): grant_type=authorization_code&code=<auth_code>
```

Example:

```
curl -X POST 'https://
idcs-4c463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/
oauth2/v1/token' \
-H 'Authorization: Basic
NTFkMDY3OTl1MmFhNGM3NDlhNzkyNzZjZjdkMjRjYTY6YjZhMzc4ZmQtZjk2Mi00MTg3
LWE2NzctYTBmMGI3NmEzODhj' \
-d
'grant_type=authorization_code&code=AgAgZTM5MGMzODVjNzY3NGZhNzgzMmEx
MTI0YmU1ZDBiOTYIABC_fyt25KEcHP9-Hmjkh9y1AAAAMPia8fMCbuzkdk10_CzsrqZ-
EtGwsI_MCPFr1Y3gt25EhrhYH65mb7vkJmPvUeKdTg'
```

Identity Domain returns the key and value for an access token:

```
{"access_token": "<access-token-value>"}
```

You can copy the access token value (the expiry of the access token is set to 7 days and refresh token's to 14 days).

 **Note:**

If one desires to create a refresh token along with access token, two additional tasks need to be performed:

- While creating OAuth client, in addition to selecting grant type Authorization Code (see step 5 in [Create an OAuth Client and Acquire a Client ID and Secret](#)), you will also have to choose **Refresh**.
- While requesting for authorization code, you will have to specify the additional scope `offline_access` along with `urn:opc:cec:all` (scopes are space separated) as shown below:

```
https://  
idcs-4c463010b1bb495d8db89fd05ebeld99.identity.dev99.testdev.c  
om/oauth2/v1/authorize?  
client_id=51d06799e2aa4c749a79276cf7d24ca7&response_type=code&  
redirect_uri=https://p2qaextradplsrvcl-  
qucm2019927ac99.cec.ocp.oc-test.com/documents&scope=https://  
50B904E53A6F4EA1B45F30E225E6C99B.cec.ocp.oc-test.com:443/  
urn:opc:cec:all offline_access&nonce=12345&state=12345
```

Then using this authorization code in `/oauth2/v1/token` will return refresh token along with access token.

Use the Access Token to Access the Oracle Content Management Resource

Once the access token is acquired, call the Oracle Content Management endpoint to access the resource.

A sample curl command follows.

```
curl -X GET <Oracle-Content-Management-URL>/content/management/api/v1.1/  
channels \  
-H 'Authorization: Bearer <Access Token acquired from previous step>\' \  
-H 'Content-Type: application/json'
```

Access OCM Using Resource Owner (Identity Domain)

In this flow, the resource owner credentials are exposed to the client. The credentials are passed in `/oauth2/v1/token` API to get the token so it is important that the resource owner has a trust relationship with the client making this API call as the client must discard the password after obtaining the access token.

As this is a user token, the user requesting the token needs to have relevant OCM roles as the user's access to the OCM objects will be checked against assigned roles.

To configure a confidential application to authorize a Resource Owner grant:

1. Sign in to the Oracle Cloud admin console: <https://cloud.oracle.com>

2. Click to add a new application in the **Applications** section.
3. Select **Confidential Application**. Confidential Applications use OAuth 2.0 and can protect their OAuth client ID and client secret. To know more about Confidential Applications, see [Add a Confidential Application](#).
4. Give the application a name and, optionally, a description, and click **Next**.
5. Choose to configure this application as a client.
6. Select the **Resource Owner** grant type on the **Authorization** screen.
7. Scroll down to the **Token Issuance Policy** section and, under **Resources**, click **Add Scope** to give the application access to the Oracle Content Management instance required.
8. Select the correct instance.
9. Click the right arrow to select the scope. The one ending in `urn:opc:cec:all` is the only scope required. Select the checkbox next to it and then click **Add** (note down the scope string as it will be used later on while requesting the access token).
10. Click **Next** until the end of the train. The **Resources**, **Web Tier Policy**, and **Authorization** stops are related to applications that have some resource authenticated or authorized by Identity Domain; for example, a web application. In the case of a simple server-server client, which is discussed here, this isn't relevant.
11. Click **Finish**.
12. Note the **Client ID** and **Client Secret** values because you'll need those to get a token later.
13. Check **Activate** and then click **Save** to enable the application.

To request a token:

1. Make a form post to `<domain URL>`. For Domain URL, see the figure given below.

The screenshot shows a web interface with two tabs: 'Domain information' (selected) and 'Tags'. The 'Domain information' tab displays the following details:

- OCID:** ...znsy5q [Show](#) [Copy](#)
- Domain type:** Free
- Description:** Default domain [Copy](#)
- Domain replication:** -
- Home region:** Canada Southeast (Toronto)
- Created:** Thu, Aug 26, 2021, 16:13:50 UTC
- Show domain on login:** On
- Domain URL:** ...ud.com:443 [Show](#) [Copy](#)
- Status:** ● Active

2. The following table describes the fields.

Field	Value
grant_type	password
scope	A scope you added previously, in step 9; for example: <code>https://1DF8AB52D0FF48F6992EEA3A5715B66F.ec.dev.ocp.octest.com:443/urn:opc:cec:all</code>
username	The user name to generate the token for.

Field	Value
password	The password for the user name in the preceding field.

```
POST /oauth2/v1/token HTTP/1.1
Host: <iam-domain-host>
Authorization: Basic
NTZkZWJjY2EzYjc0NDRlMWFhNjg4OGQ0ZTYzY2Y1M2Y6NDYxYzY5YjctMzJiZC00NGE0LTk4NTc
tN
WM1NzAyMWMzNDg4
Accept: */*
Content-Type: application/x-www-form-urlencoded
Content-Length: 162
grant_type=password&scope=https%3A%2F%2F1DF8AB52D0FF48F6992EEA3A5715B66F.cec.
dev.ocp.octest.
com%3A443%2Furn%3Aopc%3Acec%3Aall&username=<user-name>&password=<password>
```

This results in JSON text where the token is the value of the `access_token` field:

```
{
  "access_token":
"eyJ4NXQjUzI1NiI6IkhvRktIMFFGeHR1UDkxLWg3QlJKSUFDMU50V2R...HUQmto_oELyjRaB
p
qhI75hQJYLWRKm6ozPS57tR1EYHmWABgYw_XALMT1kMuIuRxpGB2ozngpajzNNBBu2qtKg10-
RzBTulKaxD25vKK1rznQ3p_XAOLK4CUUM-uG_PUOk49-
JDgJjuSI74hLC1kagIIM93A2jUG3g3gdUpUCZPg",
  "token_type": "Bearer",
  "expires_in": 604800
}
```

The token expiration time is given in seconds and is typically 7 days.

To use the token and access REST API endpoints, use the Bearer Authorization header as before.

 **Note:**

If one desires to create a refresh token along with access token, two additional tasks need to be performed:

1. While creating OAuth client, in addition to selecting grant type Resource Owner (See step 6 in [Create an OAuth Client and Acquire a Client ID and Secret](#)), you will also have to choose **Refresh Token**.
2. While requesting for token, you will have to specify additional scope `offline_access` along with `urn:opc:cec:all` (scopes are space separated) as shown below:

```
curl -X POST 'https://
idcs-4c463010b1bb495d8db89fd05ebeld99.identity.dev99.testdev
.com/oauth2/v1/token' \
-H 'Authorization: Basic
NTFkMDY3OTl1MmFhNGM3NDlhNzkyNzZjZjdkMjRjYTY6YjZhMzc4ZmQtZjk2
Mi00MTg3LWE2NzctYTBMGI3NmEzODhj' \
-d 'grant_type=password&scope=https://
50B904E53A6F4EA1B45F30E225EA99B.cec.ocp.oc-test.com:443/
urn:opc:cec:all
offline_access&username=John.Doe@test.com&password=johndoeпа
ssword#3'
```

This will return a refresh token along with access token.

Cloud Account Using Oracle Identity Cloud Service

The following sections describe how to use your cloud account:

- [Access OCM Using Client Credentials \(Two-Legged OAuth Flow\)](#)
- [Access OCM Using Authorization Code \(Three-Legged OAuth Flow\)](#)
- [Access OCM Using Resource Owner](#)

Access OCM Using Client Credentials (Two-Legged OAuth Flow)

For this OAuth flow, the token is requested for the client application and calls to Oracle Content Management (OCM) APIs are made using this client application's token. This OAuth client application needs to be added to OCM roles in Oracle Identity Cloud Service (IDCS) to have appropriate OCM access.

The **Identity Domain Administrator** or **Application Administrator** of the tenant creates an OAuth client for the developer with the required privileges and performs the following steps to get an access token:

1. [Create an OAuth Client and Acquire a Client ID and Secret](#)
2. [Grant the Required Oracle Content Management Roles to the Client](#)
3. [Acquire an Access Token from Oracle Identity Cloud Service \(IDCS\) for the Required Resource](#)
4. [Use the Access Token to Access the Oracle Content Management Resource](#)

Create an OAuth Client and Acquire a Client ID and Secret

The Identity Domain Administrator or an Application Administrator of the tenant creates an OAuth client for the developer with the required privileges. To do this, perform the following steps:

1. Go to the IDCS administration console:

```
https://<IDCS BaseURL>/ui/v1/adminconsole
```

2. Click to add a new application in the **Applications** section.
3. Select **Confidential Application**. The Confidential Applications use OAuth 2.0 and can protect their OAuth client ID and client secret. To know more about Confidential Applications, see [Add a Confidential Application](#).
4. Give the application a name and, optionally, a description, and click **Next**.
5. Choose to configure this application as a client.
6. Select the **client credentials** grant type on the **Authorization** screen.
7. Scroll down to the **Token Issuance Policy** section and, under **Resources**, click **add scope** to give the application access to the required Oracle Content Management instance.
8. Click the right arrow to select the scope.
The only scope required is the one ending in `urn:opc:cec:all`.
Select the checkbox next to it and then click **Add**.
9. Click **Next** until the end of the train.
The **Resources**, **Web Tier Policy**, and **Authorization** stops are related to applications that have some resource authenticated or authorized by IDCS; for example, a web application. This isn't relevant in the case of a simple server-server client, which is discussed here.
10. Click **Finish**.
11. Note the **Client ID** and **Client Secret** values because you'll need those to get a token later.
12. Check **Activate** and then click **Save** to enable the application.

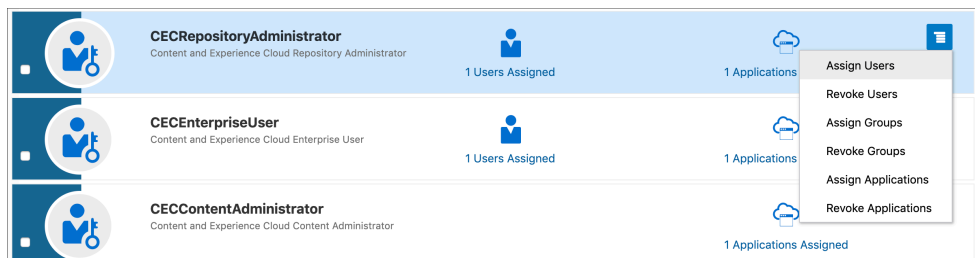
Note:

Just as a regular user, an OAuth client with the client credentials requires relevant permission on objects in your Oracle Content Management instance to be able to perform actions using REST API for Content Management. For example, it needs to be a repository member with Contributor permission to create assets in it, or a Manager on the repository to modify its settings. In the Oracle Content Management web UI, you can use the Members dialog to find your OAuth client by name and add it to the required object. A newly created OAuth client will appear in search only after the next hourly IDCS sync is complete.

Grant the Required Oracle Content Management Roles to the Client

You can grant the required roles through the Oracle Identity Cloud Service (IDCS) administration console.

1. Log in to the IDCS administration console.
2. Click **Oracle Cloud Services**.
3. Select the Oracle Content Management service instance for which you need privileges. This instance name would have been provided by your account administrator.
4. Click **Application Roles** and then right-click the burger menu on the right.
5. Choose **Assign Applications**.
6. Assign the client application to the roles based on the access privilege that is required for OCM objects. For OCM application roles, see [Application Roles](#).



7. Save the changes.

Acquire an Access Token from Oracle Identity Cloud Service (IDCS) for the Required Resource

Use the client ID and secret obtained earlier from the OAuth client to get an AccessToken for the resource.

A sample curl command to acquire an access token follows:

AccessToken Request

```
curl -X POST \
https://<idcs tenantname that is protecting this service>/oauth2/v1/
token \
-H 'Authorization: Basic <Base64 encoded clientID:clientSecret>' \
-d 'grant_type=client_credentials&scope=<scope string>'
```

For scope string, see Step 7 of [Create an OAuth Client and Acquire a Client ID and Secret](#). For Example:

```
curl -X POST 'https://
idcs-99463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/
oauth2/v1/token' \
-H 'Authorization: Basic
Y2RlNGQyMDZjYTE1NGM2Yjg2NGMyMTJiMjVmMTY3MmE6MWM0ZTAyNGQtZDU4MC00MzEzLWJ'
```

```
kZjMtZTQyMGQzMzgzZTY2' \
-d'grant_type=client_credentials&scope=https://
50B904E53A6F4EA1B45F30E225EA99B.cec.ocp.oc-test.com:443/urn:opc:cec:all'
```

IDCS returns the key and value for an access token (access token expiry is set to 7 days):

```
{"access_token":"<access-token-value>"}
```

You can copy the access token value.



Note:

The refresh token is not included in the Client Credentials grant type flow.

Use the Access Token to Access the Oracle Content Management Resource

Once the access token is acquired, call the Oracle Content Management endpoint to access the resource.

A sample curl command follows.

```
curl -X GET <Oracle-Content-Management-URL>/sites/management/api/v1/sites \
-H 'Authorization: Bearer <Access Token acquired from previous step>' \
-H 'Content-Type: application/json'
```

Access OCM Using Authorization Code (Three-Legged OAuth Flow)

In this flow, a client application is created to acquire the client ID and secret, a client browser uses these and redirects the user to IDCS to login with their username and password and on successful authentication, an authorization code is sent on the redirect URL. An access token is requested from IDCS using this authorization code. Thus, the Resource owner credentials are never exposed to the client in this flow.

As this is a user token, the user requesting the token needs to have relevant OCM roles as the user's access to the OCM objects will be checked against assigned roles.

The **Identity Domain Administrator** performs the following steps to get an access token:

1. [Create an OAuth Client and Acquire a Client ID and Secret](#)
2. [Grant the Required Oracle Content Management Roles to the Client](#)
3. [Acquire an Access Token from Oracle Identity Cloud Service \(IDCS\) for the Required Resource](#)
4. [Use the Access Token to Access the Oracle Content Management Resource](#)

Create an OAuth Client and Acquire a Client ID and Secret

The Identity Domain Administrator or an Application Administrator of the tenant creates an OAuth client for the developer with the required privileges. To do this, perform the following steps:

1. Go to the IDCS administration console:

`https://<IDCS BaseURL>/ui/v1/adminconsole`

2. Click to add a new application in the **Applications** section.
3. Select **Confidential Application**. The Confidential Applications use OAuth 2.0 and can protect their OAuth client ID and client secret. To know more about Confidential Applications, see [Add a Confidential Application](#).
4. Give the application a name and, optionally, a description, and click **Next**.
5. Choose to configure this application as a client.
6. Select the **Authorization code** grant type on the **Authorization** screen.
7. Enter a **redirect URL** value to point to the URL where the user needs to be redirected after authorization. It can be any URL pointing to customer's application.
8. Enter a **Post logout URL** value to point to the Oracle Content Management service administration console.

 **Note:**

It is not a mandatory field while creating a client application with the Authorization Code grant type. As mentioned in the IDCS documentation (see [Using OpenID Connect for Log Out](#)), this could be again any client-specific URL to where the redirection happens after logout.

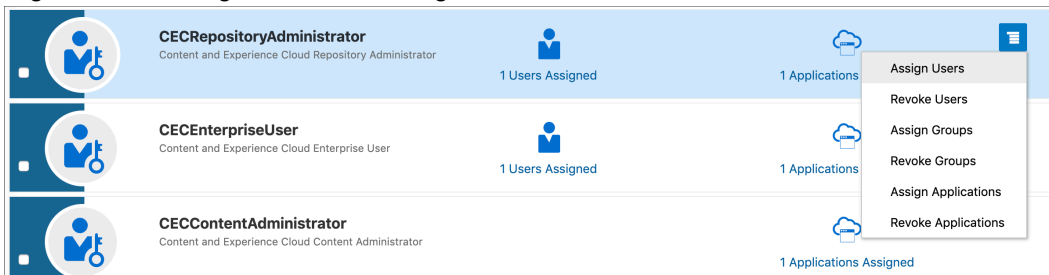
9. Scroll down to the **Token Issuance Policy** section and, under **Resources**, click **add scope** to give the application access to the required Oracle Content Management instance.
10. Click the right arrow to select the scope.
The only scope required is the one ending in `urn:opc:cec:all`.
Select the checkbox next to it and then click **Add** (note down the scope string as it will be used later on while requesting the access token).
11. Click **Next** until the end of the train.
The **Resources**, **Web Tier Policy**, and **Authorization** stops are related to applications that have some resource authenticated or authorized by IDCS; for example, a web application. This isn't relevant in the case of a simple server-server client, which is discussed here.
12. Click **Finish**.
13. Note the **Client ID** and **Client Secret** values because you'll need those to get a token later.
14. Check **Activate** and then click **Save** to enable the application.

The client can store the **Client ID** and **Client Secret** values securely in a credential store.

Grant the Required Oracle Content Management Roles to the Client

You can grant the required OCM roles to the user through the Oracle Identity Cloud Service (IDCS) administration console.

1. Log in to the IDCS administration console.
2. Click **Oracle Cloud Services**.
3. Select the Oracle Content Management service instance for which you need privileges. This instance name would have been provided by your account administrator.
4. Click **Application Roles**.
5. Right-click the burger menu on the right.



6. Choose **Assign Users**.
7. Assign the user to the roles based on the access level that is required for OCM objects. For information about OCM application roles, see [Application Roles](#).
8. Save the changes.

Acquire an Access Token from Oracle Identity Cloud Service (IDCS) for the Required Resource

Acquisition of the access token consists of multiple steps:

1. The UI of the customer's application makes authorization code requests for the user to IDCS.
 - a. The UI makes a GET call to IDCS with the URL given below. No authorization is required (this needs to be accessed via browser as this will open IDCS login UI, Postman will not work in this case).

```
https://<IDCS-service-instance>/oauth2/v1/authorize?
client_id=<clientid>&response_type=code&redirect_uri=<client-redirect-
uri>&scope=<scope string from Step 9 of Create oauth
client>&nonce=<nonce-value>&state=1234
```

Where nonce-value and state are optional.

Example:

```
https://
idcs-4c463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/
oauth2/v1/authorize?
client_id=51d06799e2aa4c749a79276cf7d24ca7&response_type=code&redirect
_uri=https://p2qaextradp1srvcl-qucm2019927ac99.cec.ocp.oc-test.com/
documents&scope=https://50B904E53A6F4EA1B45F30E225E6C99B.cec.ocp.oc-
test.com:443/urn:opc:cec:all&nonce=12345&state=12345
```

- b. IDCS shows the login UI.

- c. The user logs in to the login UI. IDCS redirects to redirect URL with the Authorization code as shown below:

```
<redirection_url>?code=<auth_code>&state=12345
```

Example:

```
https://p2qaextradplsrvcl-qucm2019927ac99.cec.ocp.oc-test.com/  
documents?  
code=AgAgZTM5MGMzODVjNzY3NGZhNzgzMmExMTI0YmU1ZDBiOTYIABC_fyt25KEc  
HP9-Hmjkh9y1AAAAMPia8fMCbuzkdk10_CzsrqZ-  
EtGwsI_MCPFr1Y3gt25EhrhYH65mb7vkJmPvUeKdTg&state=12345
```

2. The UI makes a call to IDCS to get the user token, passing `auth_code` obtained in the previous step with the following payload:

```
Authorization: Basic <Base64 encoded clientID:clientSecret>  
Method: Post  
URL: https://<IDCS-service-instance>/oauth2/v1/token  
BODY (as form data): grant_type=authorization_code&code=<auth_code>
```

Example:

```
curl -X POST 'https://  
idcs-4c463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/  
oauth2/v1/token' \  
-H 'Authorization: Basic  
NTFkMDY3OTl1MmFhNGM3NDlhNzkyNzZjZjZkdMjRjYTY6YjZhMzc4ZmQtZjk2Mi00MTg3  
LWE2NzctYTBmMGI3NmEzODhj' \  
-d  
'grant_type=authorization_code&code=AgAgZTM5MGMzODVjNzY3NGZhNzgzMmEx  
MTI0YmU1ZDBiOTYIABC_fyt25KEcHP9-Hmjkh9y1AAAAMPia8fMCbuzkdk10_CzsrqZ-  
EtGwsI_MCPFr1Y3gt25EhrhYH65mb7vkJmPvUeKdTg'
```

IDCS returns the key and value for an access token:

```
{"access_token": "<access-token-value>"}
```

You can copy the access token value (the expiry of the access token is set to 7 days and refresh token is set to 14 days).

 **Note:**

If one desires to create a refresh token along with an access token, two additional tasks need to be performed:

- While creating an OAuth client, in addition to selecting grant type Authorization Code (see step 5 in [Create an OAuth Client and Acquire a Client ID and Secret](#)), you will also have to choose **Refresh**.
- While requesting for authorization code, you will have to specify the additional scope `offline_access` along with `urn:opc:cec:all` (scopes are space separated) as shown below:

```
https://
idcs-4c463010b1bb495d8db89fd05ebeld99.identity.dev99.testdev.c
om/oauth2/v1/authorize?
client_id=51d06799e2aa4c749a79276cf7d24ca7&response_type=code&
redirect_uri=https://p2qaextradplsrvcl-
qucm2019927ac99.cec.ocp.oc-test.com/documents&scope=https://
50B904E53A6F4EA1B45F30E225E6C99B.cec.ocp.oc-test.com:443/
urn:opc:cec:all offline_access&nonce=12345&state=12345
```

Then using this authorization code in `/oauth2/v1/token` will return refresh token along with access token.

Use the Access Token to Access the Oracle Content Management Resource

Once the access token is acquired, call the Oracle Content Management endpoint to access the resource.

A sample curl command follows.

```
curl -X GET <Oracle-Content-Management-URL>/content/management/api/v1.1/
channels \
-H 'Authorization: Bearer <Access Token acquired from previous step>' \
-H 'Content-Type: application/json'
```

Access OCM Using Resource Owner

In this flow, the resource owner credentials are exposed to the client. The credentials are passed in `/oauth2/v1/token` API to get the token. So it is important that the resource owner has a trust relationship with the client making this API call, since the client must discard the password after obtaining the access token.

Because this is a user token, the user requesting the token needs to have relevant OCM roles as the user's access to the OCM objects will be checked against assigned roles.

To configure a confidential application to authorize a Resource Owner grant:

1. Go to the IDCS admin console: `https://<IDCS BaseURL>/ui/v1/adminconsole`
2. Click to add a new application in the **Applications** section.

3. Select **Confidential Application**. Confidential Applications use OAuth 2.0 and can protect their OAuth client ID and client secret. To know more about Confidential Applications, see [Add a Confidential Application](#).
4. Give the application a name and, optionally, a description, and click **Next**.
5. Choose to configure this application as a client.
6. Select the **Resource Owner** grant type on the **Authorization** screen.
7. Scroll down to the **Token Issuance Policy** section and, under **Resources**, click **Add Scope** to give the application access to the Oracle Content Management instance required.
8. Select the correct instance.
9. Click the right arrow to select the scope. The one ending in `urn:opc:cec:all` is the only scope required. Select the checkbox next to it and then click **Add** (note down the scope string as it will be used later on while requesting the access token).
10. Click **Next** until the end of the train. The **Resources**, **Web Tier Policy**, and **Authorization** stops are related to applications that have some resource authenticated or authorized by IDCS; for example, a web application. In the case of a simple server-server client, which is discussed here, this isn't relevant.
11. Click **Finish**.
12. Note the **Client ID** and **Client Secret** values because you'll need those to get a token later.
13. Check **Activate** and then click **Save** to enable the application.

To request a token:

1. Get the token by making POST to `<idcs-base-url>/oauth2/v1/token` using client application's client ID and secret with basic auth. Following is the payload:

```
Authorization: Basic <Base64 encoded clientID:clientSecret>
Method: Post
URL: <idcs-base-url>/oauth2/v1/token
BODY (as form data): grant_type=password&scope==<scope string from
Step 7 above>&username=<user name>&password=<password>
```

2. The following table describes the fields.

Field	Value
grant_type	password
scope	A scope you added previously, in step 7; for example: <code>https://1DF8AB52D0FF48F6992EEA3A5715B66F.ec.dev.ocp.octest.com:443/urn:opc:cec:all</code>
username	The user name to generate the token for.
password	The password for the user name in the preceding field.

Example:

```
curl -X POST 'https://
ids-4c463010b1bb495d8db89fd05ebe1d99.identity.dev99.testdev.com/
oauth2/v1/token' \
-H 'Authorization: Basic
NTFkMDY3OTllMmFhNGM3NDlhNzkyNzZjZjdkMjRjYTY6YjZhMzc4ZmQtZjk2Mi00MTg3LWE2Nz
ctYTBmMGI3NmEzODhj' \
-d 'grant_type=password&scope=https://
50B904E53A6F4EA1B45F30E225EA99B.cec.ocp.oc-test.com:443/
urn:opc:cec:all&username=John.Doe@test.com&password=johndoepassword#3'
```

This results in JSON text where the token is the value of the `access_token` field:

```
{
  "access_token":
"eyJ4NXQjUzI1NiI6IkhvRktIMFFGeHR1UDkxLWg3QlJKSUFDMU50V2R...HUQmto_oELyjRaB
p
qh
I75hQJYLWRKm6ozPS57tR1EYHmWABgYw_XALMT1kMuIuRxpGB2ozngpajzNNBBu2qtKg10-
RzBTulKaxD25vKK1rznQ3p_XAOLK4CUUM-uG_PUOk49-
JDgJjuSI74hLC1kagI1M93A2jUG3g3gdUpUCZPg",
  "token_type": "Bearer",
  "expires_in": 604800
}
```

The token expiration time is given in seconds and is typically 7 days.

To use the token and access REST API endpoints, use the Bearer Authorization as before.

 **Note:**

If one desires to create a refresh token along with access token, two additional tasks need to be performed:

1. While creating OAuth client, in addition to selecting grant type Resource Owner (See step 6 in [Create an OAuth Client and Acquire a Client ID and Secret](#)), you will also have to choose **Refresh Token**.
2. While requesting for token, you will have to specify additional scope `offline_access` along with `urn:opc:cec:all` (scopes are space separated) as shown below:

```
curl -X POST 'https://  
idcs-4c463010b1bb495d8db89fd05ebeld99.identity.dev99.testdev  
.com/oauth2/v1/token' \  
-H 'Authorization: Basic  
NTFkMDY3OTl1MmFhNGM3NDlhNzkyNzZjZjdkMjRjYTY6YjZhMzc4ZmQtZjk2  
Mi00MTg3LWE2NzctYTBmMGI3NmEzODhj' \  
-d 'grant_type=password&scope=https://  
50B904E53A6F4EA1B45F30E225EA99B.cec.ocp.oc-test.com:443/  
urn:opc:cec:all  
offline_access&username=John.Doe@test.com&password=johndoeпа  
ssword#3'
```

This will return a refresh token along with access token.

Download the Swagger File for a REST API

Download a REST API Swagger file for use in your development project.

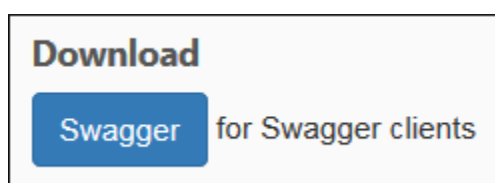
The Swagger file for each REST API is part of the published REST API document. You can download it from the left navigation tree.

For example, to download and copy the Swagger file for the *REST API for Documents*:

1. On `docs.oracle.com`, open the REST API document at [REST API for Documents](#).
2. On the left, click the download symbol:



3. Click the Swagger button:



The text from the Swagger file for the REST API is displayed.

```
{
  "swagger": "2.0",
  "info": {
    "title": "REST API for Content Management 1.2",
    "version": "2017.07.04",
    "description": "The REST API for Content Management 1.2
other sites.",
    "x-summary": "The REST API for Content Management 1.2 en
other sites."
  },
  "schemes": [
    "https"
  ],
  "produces": [
    "application/json",
    "application/xml"
  ],
  "consumes": [
    "application/json",
    "application/xml"
  ],
  "paths": {
    "/documents/api/1.2/applinks/folder/{folderId}": {
      "post": {
        "summary": "Create Folder Applink",
```

4. From the **Edit** menu, choose **Select All**, and then choose **Copy**.
5. Paste the copied text into a text file.

REST API for Activity Log

You can use Oracle Cloud REST API for Activity Log to search activities in Oracle Content Management.

The [REST API for Activity Log](#) has several categories of endpoints, which are described in the following table.

Category	Description
Audit Log	Provides the details of activities and its related data.
Events	Provides the details of types and categories.

REST API for Content Capture

You can use the Oracle Cloud REST API for Content Capture for the manipulation of documents when they are locked to an external processor job step. This includes changing a document's metadata field values, adding or removing attachments to documents, and even deleting documents.

The [REST API for Content Capture](#) has several categories of endpoints, which are described in the following table.

Category	Description
Attachments	Manages document attachments in Capture
Auditing	Queries and analyzes batch and document processing
Batches	Manages batches in Content Capture
Documents	Manages documents in Capture
Steps	Queries and updates processing task queues
System	Provides system-level operations

REST API for Content Delivery

You can use the Oracle Cloud REST API for Content Delivery to fetch things from channels in an asset repository.

The [REST API for Content Delivery](#) has several categories of endpoints, which are described in the following table.

Category	Description
AutoSuggestions	Use the AutoSuggestions resource to suggest item keywords for auto-completion of a default search.
Item	Use the Item resource to get published items, previews of items, item metadata, or taxonomies of items. Use the Published Item resource to get the metadata catalog preview of an item.
Item Variations	Use the Item Variations resource to get item variations, a content item for item variations, and item variations by variation type.
Items	Use the Items resource to search published items or get the metadata catalog of published items.
Items by slug	Use the Items by slug resource to manage items by slug and to provide details about the metadata catalog, preview, taxonomies, published information, and variations of an item.
Provider Tokens	Use the Provider Tokens resource to generate a provider token.
Published Item	Use the operations from the Published Item category.
Recommendations	Use the Recommendations resource to access published recommendation results.
Renditions	Use the Renditions resource to get a digital assets file and metadata catalog, to get metadata for digital assets or renditions, or to get information about published assets or renditions.
Renditions by slug	Use the Renditions by slug resource to get the published digital assets native file by slug, to get the published native resource by slug using a file name, to get a published rendition by slug, or to get a published rendition by slug using a file name.
Taxonomies	Use the Taxonomies resource to get the metadata of a category, published taxonomies, or published categories, to list all taxonomies, to read a published category or taxonomy, or to search published categories.
Version Catalog	Use the Version Catalog resource to get information about APIs, API versions, or API metadata.

REST API for Content Management

You can use the Oracle Cloud REST API for Content Management to manage assets in Oracle Content Management. Assets include content items as well as digital assets and their renditions.

The [REST API for Content Management](#) has several categories of endpoints, which are described in the following table.

Category	Description
Archive	Use the Archive resource to cancel, schedule, delete, list, and read the archived assets.
Archive Bulk Operations	Use the Archive Bulk Operations resource to manage bulk operations on archived assets.
Archive Items Search	Use the Archive Items Search resource to manage archived items search.
Asset Refresh Bulk Operations	Use the Asset Refresh Bulk Operations resource to manage bulk operations for asset refresh.
Assign Primary Channel to Asset Bulk Operations	Use the resource to manage bulk operations for assigning primary channel.
AutoSuggestions	Use the AutoSuggestions resource to suggest item keywords for auto-completion of a default search.
Channel Secret	Use the Channel Secret resource to generate, refresh, or delete a channel secret.
Channels	Use the Channels resource to create, delete, read, or update a channel, to list all channels, or to list all permissions on a channel.
Collections	Use the Collections resource to create, delete, read, or update a collection, to list all collections in a repository, or to list all permissions on a collection.
Connectors	Use the Connectors resource to list all connectors.
Digital Item Renditions	Use the Digital Item Renditions resource to get a rendition of a digital item or a digital item native file with or without a file name.
Digital Item Renditions by Slug	Use the Digital Item Renditions resource to get a rendition of a digital item by slug.
Editorial Roles	Use the editorial roles resource to create, copy, delete, read, update, list all permissions, list editorial roles, and list editorial role referenced by repositories.
File Extensions	Use the File Extensions resource to list file extensions or to read a file extension.
Item Revisions	Use the Item Revisions resource to list item revisions, list item revisions by slug, read an item revision, or read an item revision by slug.
Item Variations	Use the Item Variations resource to list all item variations of a variation type, read an item or items by variation type, or update the master item of an item variations set.

Category	Description
Items	<ul style="list-style-type: none"> • Create or delete a content item or digital item • Generate taxonomy suggestions for content items on demand • List all suggested taxonomies and categories or all taxonomies and categories of an item • Get the workflow instance details for an item • List the time zones, channels, collections, published channels, relationships, tags, or variations of an item, • Read the lock, publish, or version information of an item • Read the on-demand taxonomy suggestion and operations status or the workflows of an item • Submit an item to workflow, take action on a workflow task, update a digital item with a new file, update an item, or update the lock status of an item.
Items Bulk Downloads	Use the Items Bulk Operations resource to perform bulk items downloads.
Items Bulk Operations	Use the Items Bulk Operations resource to perform bulk items operations, read item operations or status, or publish item IDs.
Items by Slug	Use the Items by Slug resource to manage items by slug: <ul style="list-style-type: none"> • List all item variations of a variation type • List an item's channels, collections, lock information, permissions, publish information, published channels, relationships, tags, taxonomies, variations, version information, or workflow information. • Read an item or an item variation of a variation type value by slug, • Read the master of an item variation set by slug
Items Search	Use the Items Search resource to manage items search queries, get the job status for similar items, query items, or query similar items.
Language Codes	Use the Language Codes resource to create a custom language code, delete a language code, list all valid language codes, read a language code, or update a language code.
Languages	Use the Languages resource to list the names of all known language codes.
Localization Policies	Use the Localization Policies resource to create, delete, read, or update a localization policy or to list all localization policies.
Taxonomies	Use the Taxonomies resource to manage your content: <ul style="list-style-type: none"> • Create, update, or delete a taxonomy • Get, create, update, copy, or delete a category in a taxonomy • Create, copy, update, promote, read, publish, unpublish, or delete a taxonomy • List all taxonomies or list all categories in a taxonomy • Copy, create, or delete a category • Read a taxonomy, a category, or the copy category, draft creation, the promote status • Create a new draft version • Read the promote, publish, or unpublish job status • Search categories • Update a category's properties, including moving it in the tree <p>After taxonomies are promoted, they can be assigned to repositories. After taxonomies are assigned to repositories, users can apply categories to assets.</p>
OAuth Tokens	Use the OAuth Tokens resource to generate an OAuth token.
Permission Operations	Use the Permission Operations resource to perform permission operations on a resource or to read permission operations status.

Category	Description
Permission Sets	Use the Permission Operations resource to create, read, convert, delete, list, and update a permission set.
Provider Tokens	Generate a provider token for an asset for a specific version.
Recommendations	Use the Recommendations resource to manage recommendations: <ul style="list-style-type: none"> • Create, delete, list, update, read, and publish recommendations • Read a recommendation's published and unpublished item IDs • Approve or reject a Recommendation • Create, delete, list, read, and update audience attributes
Repositories	Use the Repositories resource to create, delete, read, or update a repository, to list all permissions on a repository, or to list all repositories.
Scheduled Jobs	Use the Scheduled Jobs resource to create, list, read, and update a scheduled publish job.
Tokens	Use the Tokens resource to read a Cross-Site Request Forgery (CSRF) valid token.
Types	Use the Types resource to create, delete, read, or update a type or to list all types, all data types, or all permissions on a type.
Workflow Roles	Use the Workflow Roles resource to add or remove members of a process role, get members or details of a process role, or list the roles of all registered workflows.
Workflow Tasks	Use the Workflow Tasks resource to list workflow tasks assigned to the current user, or to read a workflow task.
Workflows	Use the Workflows resource to reregister or deregister a workflow, list all workflows, list all permissions on a workflow, read a workflow, or update a workflow.

REST API for Content Preview

You can use the Oracle Cloud REST API for Content Preview to preview items, item variations, items by slug, renditions, and renditions by slug.

The [REST API for Content Preview](#) has several categories of endpoints, which are described in the following table.

Category	Description
Item	Use the Item resource to preview the latest management version of items. An item is uniquely identified by an ID.
Item Variations	Use the Item Variations resource to preview the list of item variations, item variations by variation type, and a specific variation.
Items	Use the Items resource to search across the preview of items.
Items by Slug	Use the Items by Slug resource to manage items by slug and to preview the items, their variations, and renditions by slug. An item is uniquely identified by its slug.
Recommendations	The resources used to preview the recommendation results.
Renditions	Use the Renditions resource to preview the latest version of digital assets native file and renditions.

Category	Description
Renditions by Slug	Use the Renditions by Slug resource to preview the latest version of the digital assets native file and its renditions by slug.
Taxonomies	The resources used to preview taxonomies and their respective categories.
Version Catalog	The Version Catalog resource returns information about the available versions of the API.

REST API for Conversations

You can use the Oracle Cloud REST API for Conversations to create and manage conversations in your cloud resources that enable real-time collaboration between individuals and teams, and connect your business processes, enterprise applications, and content.

The [REST API for Conversations](#) has several categories of endpoints, which are described in the following table.

Category	Description
Collaboration	Enables collaboration among users through conversations.
Configuration	Configures resources and services.
Security	Manages connections and access.
Social	Marks favorite conversations and documents, and collaborate with people.

REST API for Documents

You can use the Oracle Cloud REST API for Documents to create client applications that interact with folders and files stored on an Oracle Content Management server.

The [REST API for Documents](#) has several categories of endpoints, which are described in the following table.

Category	Description
Applinks	Provides basic applink operations.
Catalog	Provides the information about what resources are available for a particular version.
Client Applications	Provides the operations from the Client Applications category.
Configuration	Provides basic configuration operations.
Files	Provides basic file operations.
Folders	Provides basic folder operations.
Metadata Collection	Provides the operations from the Metadata Collection category.
Publiclinks	Provides the basic public links operations
Shares	Provides basic sharing operations.

Category	Description
Sites	Provides basic site operations.
Templates	Provides basic template operations.
Users	Provides the basic information about users to identify them for folder and file sharing purposes.

REST API for Self-Management

You can use the Oracle Cloud REST API for Self-Management to view authenticated user details.

The [REST API for Self-Management](#) has one category of endpoints for you to view user details, which is described in the following table.

Category	Description
Self-Management	Provides the ability to view the authenticated user details in Oracle Content Management.

REST API for Sites Management

You can use the Oracle Cloud REST API for Sites Management provides to create sites from templates and then manage the life cycle of those sites in Oracle Content Management.

The [REST API for Sites Management](#) has several categories of endpoints, which are described in the following table.

Category	Description
Components	Import, export, list, read, copy, update, publish, share, delete, restore, and share components.
Policies	Manage policies and manage the membership of policies with access lists. You can read and edit policies associated with site management operations and control who can perform site management operations using access lists.
Requests	Requests are made when creating a site. A site request is rejected or approved through a review. You can list requests relating to sites edit requests that have been rejected, and retry failed requests. Also, you can view reviews for site requests. Approve or reject requests for new sites or requests to copy sites.
Settings	Get and update site management settings.
Sites	Create, list, read, copy, share, delete, restore, publish, activate, and deactivate sites. Get the progress of site-related jobs and information such as the associated repository and the user that created or last modified a resource. List sites and requests of a site.
Templates	Import, export, list, read, copy, share, delete, and restore templates. List site templates and images. Get the progress of template-related jobs. You can also create policies and associate them with a template to be used when creating a site, or delete policies associated with templates. You can use policies to enforce site-creation approval, who can use a template, and what security the site will have.
Themes	Import, list, read, copy, publish, update, share, delete, and restore themes.

REST API for Users and Groups

You can use the Oracle Cloud REST API for Users and Groups to manage users, groups, one-on-one conversations, and pictures.

The [REST API for Users and Groups](#) has one category of endpoints for you to manage users and groups of users, which is described in the following table.

Category	Description
People and Groups	Enables you to manage users and groups of users.

REST API for Webhooks Management

You can use the Oracle Cloud [REST API for Webhooks Management](#) to manage webhooks in Oracle Content Management.

The REST API for Webhooks Management has several categories of endpoints, which are described in the following table.

Category	Description
CSRF Token	Returns a CSRF token.
Webhooks	Create, delete, read, update, list logs, and list all webhooks.

Use REST APIs for Content Search

Oracle Content Management REST APIs allow you to search published or managed content (content items and digital assets) by fetching items matching a search query submitted to the `/items` resource.

Search Query Operators

Search Query Scopes

- `/content/published/api/v1.1/items`
The search query scope in the *REST API for Content Delivery* is a publishing target which is specified by its channel token. A channel token must be provided as either a query parameter or a request header.
- `/content/management/api/v1.1/items`
The search query scope is controlled by editorial permissions granted to the API client users. A search query will fetch only items that users can see based on the repository membership and granular editorial permissions on assets in it.

If the publishing target that you selected for a request to the REST API for Content Delivery is secure (i.e., assets are published to a secure publishing channel), the search request that you send requires using OAuth for authorization. Any search request that you send to the [REST API for Content Management](#) requires using OAuth for authorization.

Search Query Expressions

Item searches in the REST API for Content Management and the REST API for Content Delivery share query parameters and use the same syntax for building a search query expression. Therefore, a single term, the Search API is used below to refer to item searches in either REST API.

The Search API allows you to specify a search query expression as a value of the `q` parameter in requests submitted to the `/items` resource. The Search query expression syntax supports various operators to define match conditions (e.g., `eq`, `ne`, `co`, or `sw`) plus `AND`, `OR`, and `()` operators to combine match conditions into a complex query expression that match the field values within a single asset type or across multiple types. Extra search query request parameters allow you to control the number of matching items to be fetched, how matching items are paginated and sorted, or what data fields are returned for matching items. Keep in mind that regardless of the search scope and expression, the maximum page size that the API can return matching items is limited to 500.

Two generic forms of search queries are supported in the Search API

- Query within an asset type: Query expression requires you to specify an asset type and allows references to both standard and user-defined data fields of this type in query conditions.
For example: `/api/v1.1/items?q=(type eq "Employee" AND fields.role eq "Senior Developer")`. Here, the role is a field of type *Employee*.
- Query across asset types: Depending on how a query expression is formulated, the Search API supports two types of cross-type queries:
 - Global query on standard fields: Query expression allows you to specify more than one asset type using parentheses `()` or no type at all. Such query is treated as a global (untyped) query with the ability to reference standard fields only in query conditions.
For example: `/api/v1.1/items?q=(name eq "John" or description co "John" and (type eq "t1" or type eq "t2"))`
 - Query across specific types: Query expression allows you to specify more than one asset type using curly brackets `{}`. Such query is treated as a generic query across specified asset types with the ability to reference both standard and user-defined fields in query conditions. For more information, see [Search Across Types](#).
For example: `/api/v1.1/items?q=(name eq "John" and {type eq "t1" and fields.ud1 eq "ud1val"} or {type eq "t2" and fields.ud2 eq "ud2val"})`

The standard data fields that are supported in asset types

- In Management API:
`id`, `type`, `name`, `description`, `typeCategory`, `slug`, `translatable`, `language`, `createdBy`, `createdDate`, `updatedBy`, `updatedDate`, `repositoryId`, `channels`, `collections`, `status`, `tags`, `isPublished`, `languageIsMaster`, `taxonomies`
- In Delivery API:
`id`, `type`, `name`, `description`, `typeCategory`, `slug`, `translatable`, `language`, `createdDate`, `updatedDate`, `taxonomies`

While querying using standard fields in either search query form, the various data fields are treated differently:

Data Fields	Data Type
<code>name</code> , <code>description</code> , <code>type</code> , <code>typeCategory</code> , <code>slug</code> , <code>language</code> , <code>status</code>	text data type (single valued)

Data Fields	Data Type
id, repositoryId	reference (single valued)
createdDate, updatedDate	datetime (single valued)
translatable, isPublished, languageIsMaster	boolean (single valued)
collections, channels, tags, taxonomies, suggestedTaxonomies	reference data type (multi valued)

In either search query form, it is possible to search by matching items anywhere (i.e., matching any standard or user-defined data field) by specifying a match criteria via the query parameter `default` on request. Such a criterion is treated as a generic query that matches values of the items across all fields that are indexed as an aggregate text data. For example:

- `/api/v1.1/items?default="coffee"`
- `/api/v1.1/items?q=(type eq "Employee")&default="senior"`

Find more details about the following:

- [Search operators](#) and query parameters that are supported in the Search API.
- [Two-level deep search](#) and related options that are supported for order matching items by a reference field.
- [Search for custom fields](#) across type and related to that order by parameter.
- [Getting dynamic asset counts on a taxonomy](#) category and using this API to support faceted search.

Stop words (words to be filtered out in your search)

The following English stop words will be filtered out for default search and typed anyField search:

a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with.

Search Operators

Searches in the REST API for Content Management or REST API for Content Delivery allow you to create search query expressions using a rich set of operators to define field match conditions plus, AND, OR, and () operators to combine them into a complex query expression that can match the field values within a single asset type or across multiple types. The table below lists search query operators that are currently supported in the Search API.

Operator	Example	Supported Data Types	Description
eq	<pre>?q=(name eq "John") ?q=(type eq "DigitalAsset") ?q=(type eq "Employee" and fields.DOB eq "1994/09/26T16:23:45.208") ?q=(type eq "File" and fileextension eq "docx") ?q=(taxonomies.categories.id eq "B9F568DC43C54803AC76012783FA5101") ?q=(taxonomies.categories.name eq "cars")? ?q=(taxonomies.categories.apiName eq "cars-unique") ?q=(taxonomies.categories.nodes.id eq "B9F568DC43C54803AC76012783FA5101") ?q=(taxonomies.categories.nodes.name eq "cars")? ?q=(taxonomies.categories.nodes.apiName eq "cars") ?q=(suggestedTaxonomies.categories.id eq "B9F568DC43C54803AC76012783FA5101")</pre>	text, reference, number, decimal, boolean, datetime.	The equals operator eq matches the exact value supplied in the query. This operator is not applicable to multivalued data types. The value provided with this operator is not case-sensitive except for standard fields. This operator considers even special characters in the value.

Operator	Example	Supported Data Types	Description
ne	<code>?q=(name ne "John")</code>	text, reference, number, decimal, boolean, datetime.	The <code>not equals</code> operator <code>ne</code> matches items that do not match the exact value supplied in the query. This operator is not applicable to multivalued data types. The value provided with this operator is not case-sensitive except for standard fields. This operator considers even special characters in the value. The <code>ne</code> operator can be used with queries on user-defined fields within an asset type. But a cross-type query cannot include search terms on user-defined fields.
co	<code>?q=(type eq "Employee" AND name co "john alex")</code> <code>?q=(type eq "Car" AND fields.features co "manual")</code> ? <code>q=(taxonomies.categories.name co "red")</code> ? <code>q=(taxonomies.categories.nodes.name co "car")</code>	text, reference, number, decimal, datetime, largertext	The <code>contains</code> operator <code>co</code> matches every word given in the criteria. The words are formed by splitting the value by special characters. It gives the results that have at least one of the words (in this example, <code>john</code> or <code>alex</code> or both). This operator does not consider special characters in the value while searching. This operator does not search for stop words . This operator is applicable to <code>text</code> , <code>largertext</code> in case of single-valued attributes, whereas for multivalued attributes, it is applicable to <code>text</code> , <code>reference</code> , <code>number</code> , <code>decimal</code> , <code>datetime</code> , <code>largertext</code> . To understand the possible datetime formats, refer to Supported Date and Time Formats . The value provided with this operator is not case-sensitive.
nc	<code>?q=(name nc "john alex")</code>	text, reference, number, decimal, datetime, largertext	The <code>not contains</code> operator <code>nc</code> matches items that would not match a <code>co</code> operator with the same criteria.

Operator	Example	Supported Data Types	Description
sw	<pre>?q=(type eq "Employee" AND name sw "Joh") ?q=(type eq "Employee" AND fields.city sw "Los") ? q=(taxonomies.categories. name sw "cat") ? q=(taxonomies.categories. nodes.name sw "red")</pre>	text	The starts with operator sw matches only the initial character values given in the field condition. This operator is not applicable to multivalued data types. The value provided with this operator is not case-sensitive.
ge	<pre>?q=(type eq "Employee" AND fields.age ge "40") ?q=(type eq "DigitalAsset" AND updatedAt ge "20171026")</pre>	number, decimal, datetime	The greater than or equal to operator ge matches only numeric and datetime values. To understand the possible datetime formats, refer to Supported Date and Time Formats . This operator is not applicable to multivalued data types.
le	<pre>?q=(type eq "Employee" AND fields.weight le "60.6")</pre>	number, decimal, datetime	The less than or equal to operator le matches only numeric and datetime values. To understand the possible datetime formats, refer to Supported Date and Time Formats . This operator is not applicable to multivalued data types.
gt	<pre>?q=(type eq "Employee" AND fields.age gt "20")</pre>	number, decimal, datetime	The greater than operator gt matches only numeric and datetime values. To understand the possible datetime formats, refer to Supported Date and Time Formats . This operator is not applicable to multivalued data types.
lt	<pre>?q=(type eq "Employee" AND fields.age lt "20") ?q=(type eq "Employee" AND createdAt lt "1994/09/26T16:23:45.208")</pre>	number, decimal, datetime	The less than operator lt matches only numeric and datetime values. To understand the possible datetime formats, please refer to Supported Date and Time Formats . This operator is not applicable to multivalued data types.

Operator	Example	Supported Data Types	Description
mt	<pre>?q=(type eq "Car" AND fields.review mt "petrol 20KMPL") ?q=(type eq "Employee" AND name mt "Jo?n") ?q=(type eq "Employee" AND name mt "Jo*") ?q=(type eq "Employee" AND fields.role mt "senior*")</pre>	text, largertext	<p>The phrase query or proximity search matches operator <code>mt</code> enables you to find words that are within a specific distance to one another. Results are sorted by the best match. It is useful for searching content items when values provided in the criteria "petrol 20kmpl" of the query need to match content that may contain "petrol fuel mileage runs 20KMPL in the speed way".</p> <p>The <code>matches</code> operator can use a wildcard (<code>?</code> or <code>*</code>) to match multiple characters within the given value. This operator is applicable to both single-valued and multivalued data types. This operator does not search for stop words. The value provided with this operator is not case-sensitive. To match exact phrases, enclose search terms in double quotes. Enclosing double quotes will be treated as part of the search term in all other contexts.</p>
sm	<pre>?q=(type eq "Employee" And fields.city sm "Rome")</pre>	text, largertext	<p>This operator <code>sm</code> allows searching for values that sound like specified criteria - also called fuzzy search, which uses, by default, a maximum of two edits to match the result. "Rome" is similar to "Dome". This operator is applicable to both single-valued and multivalued data types. The value provided with this operator is not case-sensitive.</p>
AND	<pre>?q=(type eq "Employee" AND name eq "John" AND fields.age ge "40")</pre>	-NA-	<p>The <code>AND</code> operator, can be used to put an <code>AND</code> condition between multiple query conditions. This takes precedence over <code>OR</code>.</p>

Operator	Example	Supported Data Types	Description
OR	<pre>type eq "Employee" AND name eq "John" OR fields.age ge "40"</pre>	-NA-	The OR operator can be used to put an OR condition between multiple query conditions.
()	<pre>?q=(type eq "Employee" AND (name eq "John" AND fields.age ge "40")) ?q=(type eq "Employee" AND ((name eq "John" AND fields.age ge "40") OR fields.weight ge 60))</pre>	-NA-	The parenthesis, enclosing operator () can be used to group the conditions in the criteria. This takes highest precedence, followed by AND and then by OR.

Search Queries

When you are creating a search query expression, take into account the following extra details about using operators in a specific search query or with a specific data type that are provided in the table below.

Query	Description
Type specific query	<p>This query always uses only the eq operator; eq works with case-sensitive type names. Multitype query is supported, such as ?q=(type eq "Employee" OR type eq "Address"), however, it will be a query across types.</p> <p>Example: ?q=(type eq "Employee" OR type eq "DigitalAsset") is supported because of a single type search.</p>
Date query	<p>The date query is a special type of query because of various date formats associated with it. All the datetime values in the query are assumed to be in the UTC time zone only, unless the offset is added in the ISO 8901 format. The query results will always be in the UTC timezone format for all datetime fields. Fields with the data type datetime should use the operators ge, gt, le, lt for a range query and use eq for an equals match. Supported Date and Time Formats describes date / datetime formats that are accepted in a date query.</p> <p>Example: Query Products by the manufactureDate (datetime) field using the ge operator. <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(type eq "Product" AND fields.manufactureDate ge "1989-03-26")</code>. The given date format is YYYY-MM-DD</p>
Decimal values query	<p>The decimal number query is supported up to three digits after decimal points, and the rest of the digits are ignored. <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(type eq "Product" AND fields.price ge 425.3214)</code>, will return all the products where the price is greater than or equal to 425.321.</p>

Query	Description
Large text data type query	Fields with the data type <code>largetext</code> are not stored. So the query result cannot return its value. But the field value can be used in a query condition.
Generic query	<p>The generic query is a default search query, which can be used when the user can search by value only without using any field name or operators. Internally, a generic search uses the <code>co</code> operator. This query does not support wildcard characters. This query does not search for stop words.</p> <p>Example: <code>?q=(type eq "Employee")&fields=all&default=John</code>. This will return the search results for the <code>Employee</code> type where the value <code>John</code> matches the field name, description, or any user-defined fields.</p> <p>Example: <code>?q=(type eq "Article")&default=skating</code>. This will return the search results for the <code>Article</code> type where the value <code>skating</code> matches the field name, description, or any user-defined fields.</p>
Id query	<p>This is a search query that uses the <code>Id</code> attribute.</p> <p>Example: <code>?q=(type eq "Employee" AND id eq "COREAF29AC6ACA9644F9836E36C7B558F316")</code>. The value of <code>Id</code> belongs to the item being queried, here it is the <code>Id</code> of type <code>Employee</code>. It is possible to include multiple item <code>Ids</code> in the query expression.</p> <p>Example: <code>?q=(type eq "Employee" AND (id eq "COREAF29AC6ACA9644F9836E36C7B558F316" OR id eq "COREAF29AC6ACA9644F9836E36C7B558F987"))</code>.</p>
Category query	<p>This query searches items categorized with given category <code>Id</code>, category name, category <code>apiName</code>, category node <code>Id</code>, category node name, or category node <code>apiName</code> attributes.</p> <p>Example: <code>?q=(taxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489")</code> The value of <code>taxonomies.categories.id</code> belongs to the category assigned to items being queried. It is possible to include multiple category <code>Ids</code> or names in the query expression.</p> <p>Example: <code>?q=(taxonomies.categories.name co "cars" OR taxonomies.categories.name sw "red")</code></p> <p>Example: <code>?q=(taxonomies.categories.apiName eq "All-furnitures")</code> The value of <code>taxonomies.categories.apiName</code> belongs to the category assigned to items being queried. It is possible to include multiple category <code>apiName</code> values in the query expression.</p> <p>Example: <code>?q=(taxonomies.categories.nodes.id eq "9E1A79EE600C4C4BB727FE3E39E95489")</code> The value of <code>taxonomies.categories.nodes.id</code> belongs to any node of categories assigned to items being queried. It is possible to include multiple category node <code>Ids</code> or node names in the query expression.</p> <p>Example: <code>?q=(taxonomies.categories.nodes.name co "cars" OR taxonomies.categories.nodes.name sw "red")</code></p> <p>Example: <code>?q=(taxonomies.categories.nodes.apiName eq "All-furnitures" OR taxonomies.categories.nodes.apiName eq "all-accessories")</code> The value of <code>taxonomies.categories.nodes.apiName</code> belongs to any node of categories assigned to items being queried. It is possible to include multiple category node <code>apiName</code> values in the query expression.</p>

Query	Description
Suggested category query	<p>This query searches items that could be categorized into a given category with category Id, rejected status, and language. Only one category Id can be used for filtering at a time; repositoryId is mandatory when filtering on suggestedTaxonomy. The default status is <i>not rejected</i>. When no language is specified, items from all languages are returned.</p> <p>Example: ?q=(repositoryId eq "86E125F3D78B409EBF61737636599FE1" AND suggestedTaxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489") Lists all the items from all languages that are suggested for the given category Id for the given repository and <i>not rejected</i>.</p> <p>Example: ?q=(repositoryId eq "86E125F3D78B409EBF61737636599FE1" AND suggestedTaxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489" AND suggestedTaxonomies.categories.isRejected eq "false") Lists all the items from all languages that are suggested for the given category Id for the given repository and <i>not rejected</i>.</p> <p>Example: ?q=(repositoryId eq "86E125F3D78B409EBF61737636599FE1" AND suggestedTaxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489" AND suggestedTaxonomies.categories.isRejected eq "false" AND language co "en-US") Lists all the items from the given language that are suggested for the given category Id for the given repository.</p> <p>Example: ?q=(repositoryId eq "86E125F3D78B409EBF61737636599FE1" AND suggestedTaxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489" AND suggestedTaxonomies.categories.isRejected eq "true" AND language co "en-US") Lists all the items from the given language that are previously rejected for the given category Id for the given repository.</p>
Lock status query	<p>This query searches items by locked status.</p> <p>Example: ?q=lockInfo.status eq "locked" Lists all the items that are in locked status.</p> <p>Example: ?q=lockInfo.status eq "unlocked" Lists all the items that are in unlocked status.</p> <p>Example: ?q=lockInfo.lockedBy eq "userName"&fields=lockInfo Lists all the items that are locked by the user specified and displays the lock information in results.</p>
Reference field query	<p>This search query uses reference attribute.</p> <p>Example: ?q=(type eq "Employee" AND fields.department eq "COREAF29AC6ACA9644F9836E36C7B558F412"). The value of department belongs to the Id of an item of type department.</p>

Query	Description
Match text in any field query	<p>The query searches for assets within a type or across specific types. The search in any field and binary file's text is supported by using query syntax that matches an asset anywhere. This query does not search for stop words.</p> <p>Example: <code>?q=(type eq "Employee" AND (anyField co "John Smith" and fields.address.state eq "CA"))</code></p> <p>Example: <code>?q=(type eq "cdt1" AND (anyField co "John Smith" and fields.address.state eq "CA"))</code></p> <p>Example: <code>?q=(type eq "DigitalAsset" AND anyField co "John Smith")</code></p> <p>Use query operators that are supported by the query syntax: CO (contains), MT (phrase match), NC (not contains).</p>
	<div style="border-left: 2px solid #0070C0; border-right: 2px solid #0070C0; border-bottom: 2px solid #0070C0; padding: 10px; background-color: #E6F2FF;"> <p> Note:</p> <p>The MT operator cannot be used in an anyField query to search for values in a number or decimal field.</p> </div>
Match text in a binary file query	<p>The query does typed search to match the text in the binary file of the digital asset. The search in the binary file's text in the digital assets is supported by using <code>metadata.extractedText</code> parameter in the search query within a type or across specific types.</p> <p>Example: <code>?q=(type eq "File" AND metadata.extractedText co "Employee")</code></p> <p>Example: <code>?q=({type eq "cdt1" AND metadata.extractedText mt "installed"} or {type eq "cdt2" AND metadata.extractedText mt "installed"})</code></p> <p>Use query operators that are supported by the query syntax for Text data type: CO (contains), MT (phrase match), NC (not contains).</p>

Supported Date and Time Formats

The table below describes DATE and DATETIME formats that are accepted for use in a query condition that uses the date data field:

Format	Example
YYYY-MM-DD	1989-03-26
YYYY/MM/DD	1989/03/26
DD-MM-YYYY	26-03-1989
DD/MM/YYYY	26/03/1989
YYYY-MM-DD"T"hh:mm:ss	1989-03-26T18:32:38
YYYY/MM/DD"T"hh:mm:ss	1989/03/26T18:32:38
DD-MM-YYYY"T"hh:mm:ss	26-03-1989T18:32:38
DD/MM/YYYY"T"hh:mm:ss	26/03/1989T18:32:38
YYYY-MM-DD"T"hh:mm:ss.SSS	1989-03-26T18:32:38.840

Format	Example
YYYY/MM/DD"T"hh:mm:ss.SSS	1989/03/26T18:32:38.840
DD-MM-YYYY"T"hh:mm:ss.SSS	26-03-1989T18:32:38.840
DD/MM/YYYY"T"hh:mm:ss.SSS	26/03/1989T18:32:38.840
YYYYMMDD	19890326
YYYYMMDDhhmmss	19890326183238
YYYYMMDDhhmmssSSS	19880326183238840
YYYY-MM-DD"T"hh:mm:ss.SSS+/-HH:mm	1989-03-26T18:32:38.840+05:30
YYYY-MM-DD"T"hh:mm:ss+/-HH:mm	1989-03-26T18:32:38+05:30

Search with the Querytext Parameter

You can use the `querytext` parameter of the Search Folders or Files APIs in the REST API for Documents to take advantage of string search, tag search, and custom metadata field search at the same time.

The `querytext` search string is available in the Search Folders or Files and the Search Folders or Files Under Specific Folder ID endpoints, to match folder or file names and allow for tag search or custom metadata field search as well. You can use `querytext` to search an entire directory tree in your home (`self`) directory as well as shared folders.

To set up `querytext` searches with the REST APIs:

1. Create files and folders, and add tags to them for string searches.

Tags currently support only `CONTAINS`.

 - a. Plan where you will place each tag because tags are inherited from parent folders.
 - b. Set tags, add tags, or remove tags with the following APIs: Set Folder Tags, Edit Folder Tags, Set File Tags, and Edit File Tags
2. Add metadata collections.
 - a. As an administrator, create global collections (personal collections are not supported as indexed collections).
 - b. Determine which fields you will need to support the search, and call APIs in the metadata resource to index those fields.

There is a limit of 100 fields to be indexed. You cannot remove fields from the index. The search is done on Favorite shared folders first and then other shared folders, up to 100. You may want to designate some folders as Favorites before searching, to ensure better search results.
 - c. Metadata fields of an integer type cannot be searched as a number.
 - d. You can search metadata fields of a Date type in a search API from the REST API for Documents. This enables you to search for exact date matches or for a range of dates (that is, files where the date falls between a specific start date and a specific end date).
3. Build your query.
 - a. Use a search strings in the `querytext` parameter of the Search Folders or Files and the Search Folders or Files Under Specific Folder ID endpoints to search your folder and file names, tags, and indexed metadata fields.

For examples of tags and custom metadata searches, see the descriptions of the endpoints in the [REST API for Documents](#).

- b. Search queries require URL encoding of the single-quotation-mark character (') into %60. For example, `Collection1.field1<CONTAINS>'myValue'` turns into `Collection1.field1<CONTAINS>%60myValue%60`.
- c. Start with simple queries to validate that your conditions do indeed find results.
- d. You can build more complex queries by combining parentheses, <AND> clauses, and <OR> clauses.

Set Up Searches on Metadata Fields

Set up searchable metadata fields with the Metadata Collection resource of the REST API for Documents. Then you can run text searches with custom fields.

To search metadata fields:

1. Sign in to Oracle Content Management as an administrator and create a metadata collection.

See [Metadata Collection Resource](#).

2. As an administrator, check which metadata fields are already searchable with the Get Searchable Metadata Fields endpoint, so you can determine how many fields you can add to the search index.

This REST API call retrieves all metadata fields currently available for searching content. The result list includes all metadata fields prefixed with their respective global metadata collection for the tenant. Each tenant is limited to 200 searchable fields.

3. Use the Set Searchable Metadata Fields endpoint of the [REST API for Documents](#) to specify metadata fields that are searchable.

After you reach 200 fields, you cannot index new fields to become searchable. Currently, fields cannot be removed from the search index unless the metadata collection and/or field are deleted from the system. When a collection and/or field are deleted, all the existing metadata information previously set will be lost (not recoverable). After this, new fields will be allowed to become searchable up to the 200 fields limit.

For custom metadata searches, the REST API for Documents supports only text searches using `CONTAINS`. It does not support numeric or date searches. For example, custom metadata fields created through the web user interface are not searchable because they are numeric or date type fields.

For more information about these endpoints and examples of using them, see "Set Searchable Metadata Fields" and "Get Searchable Metadata Fields" under "Metadata Collection" in [REST API for Documents](#).

Search Request Parameters

In addition to `q` or `default` parameters, a search request submitted to the `/items` resource allows you to use other request parameters to control number of items or item fields returned in response. The tables below provide details about requests accepted by the REST API for Content Delivery and REST API for Content Management.

REST API for Content Delivery

Query Parameter	Type	Description
channelToken	string	Channel token of the publish target. A channel token must be provided as either a query parameter or a request header.
default	string	Default search query expression that matches the values of items across all fields.
fields	string	<p>The fields parameter is used to control the returned fields and values in the queried item. This parameter accepts a comma-separated list of field names or all. All the user-defined field names should be provided with the fields prefix and followed by a period (.). These fields will be returned for each queried item. As all the field names are case-sensitive, users must provide the correct field names in the search query. When fields is specified as all (case-insensitive), in case of a type-specific query, all the standard fields, and user-defined fields except targettext, json, location datatype are returned for each queried item, whereas in case of a query across all types, only all standard fields are returned. The standard field Id and type are always returned in the response and cannot be filtered out. This parameter is optional in the query and by default query result shows only Id, name, description, and type in the response. Any incorrect or invalid field name given in the query will be ignored.</p> <p>In the context of a brace style cross-type query, type-specific fields may be specified using the syntax <code>name, {typename1:fields.userdefinedfieldname1,fields.userdefinedfieldname2}, {typename2:fields.userdefinedfieldname1}</code>. In the preceding example, all items of the type typename1 will have fields - name, userdefinedfieldname1, and userdefinedfieldname2, while all items of the type typename2 will have fields - name, userdefinedfieldname1. If the cross-type query does not resolve to types referenced in the typed fields clause(s), an error will be thrown.</p> <p>Example: <code>?q=(type eq "LocationType")&fields=fields.state,fields.country,updatedDate</code></p> <p>This returns Id, type, state, country, updatedDate in the search results for a LocationType with state and country fields.</p> <p>Example: <code>?q=(type eq "LocationType")&fields=all</code></p> <p>This will return all standard fields and all user-defined fields except targettext, json, location data types</p> <p>Example: <code>?fields=all</code></p> <p>This will return only standard fields (Id, type, typeCategory, name, description, slug, language, createdDate, updatedDate, taxonomies, renditions) since the type field is not used in the search query (acts as a global query)</p> <p>Example: <code>?fields=name,createdDate</code></p> <p>This will return only standard fields (such as Id, name, createdDate, and type) since the type field is not used in the search query (acts as a Global query)</p> <p>Default Value: name,description,slug,language,translatable,createdDate,updatedDate</p>
limit	integer(int32)	Number of rows to return. The default value is 100.

Query Parameter	Type	Description
links	string	Accepts a comma-separated list of rel (relation) links, which the client requires in the results. By default, all the applicable links in a resource are included in the response. Possible values are: self, canonical, describedby, first, last, prev, next. Example: links=self,canonical will only return the links with the rel property, self, or canonical.
offset	integer(int32)	Start index of response rows. The default value is 100.

Query Parameter	Type	Description
orderBy	string	<p>The <code>orderBy</code> parameter is used to control the order (ascending or descending) of queried items. This parameter is optional in the query and by default the results are sorted in the descending order of <code>updatedAt</code>. This parameter accepts <code>featured()</code> or field names separated by a colon (:), which the user wants to sort the results and sort order. Multiple sort orders are separated by semicolon (;).</p> <p>Format: <code>orderBy={fieldName1}:{asc/desc};{fieldName2}:{asc/desc}</code></p>

 **Note:**

asc stands for ascending and des for descending. asc and des are always in lower case.

In a type specific query, field names can be: name, createdAt, updatedAt (standard fields), or user-defined fields (single-valued data types: number, decimal, datetime, text). All the user-defined field names should be provided with prefix fields and followed by a period (.). In the context of a cross-type search, this parameter can also have a typed section and takes the form

```
{typename1:fields.userdefinedfieldname1:asc|desc};
{typename2:fields.userdefinedfieldname1:asc|desc}.
```

The `orderBy` parameter only supports one order by field per set of matching braces. The sort order is optional and by default it is ascending. Any incorrect or invalid field name given in the query will be ignored. When `featured()` is given, the results are sorted by the relevance of the items to the applied filter.

The `orderBy` parameter also supports the following:

- Sorting by a two-level field. If there is a two-level sort field, then it can be the only sort field in the `orderBy` parameter.

Sorting assets using user-defined text field or an attribute in JSON data stored in JSON field (embedded content) on assets in a two-level search.

 **Note:**

The two-level sorting does not work on multiple value fields, so this does not work on multiple text fields. The two-level sorting works on any array attributes on JSON fields because JSON fields can only be a single value field.

- Sorting by taxonomy root categories and sub-categories of a root category. The additional sorting parameters are as follows:


```
taxonomies.shortName["short name of taxonomy"]:
{asc/desc};taxonomies.categories.nodes.name["short
name of taxonomy|"taxonomy
name","category1","category2","category3"]
```


Query Parameter	Type	Description
		<p>where category1, category2, category3, etc. are sub-categories of a given taxonomy "taxonomy name".</p> <ul style="list-style-type: none"> • Sorting by an attribute in JSON data stored in JSON field on assets. <p>Example: <code>orderBy=name:asc</code> Returns all the items in the ascending order of name.</p> <p>Example: <code>orderBy=updateDate:asc</code> Returns all the items in the ascending order of updateDate.</p> <p>Example: <code>orderBy=fields.age:des</code> Returns all the items in the descending order of age.</p> <p>Example: <code>orderBy=fields.age</code> Returns all the items in the ascending order of age.</p> <p>Example: <code>default=apples&orderBy=featured()</code> Returns all the items sorted by the relevance of the items to the apples.</p> <p>Example: <code>orderBy=fields.blogauthor.authorage</code> Returns all the items in the ascending order of age of the author referred by referenced field "blogauthor".</p> <p>Example: <code>orderBy=fields.blogauthor.fields.authorage&twolvl_v1_1=true</code> Returns all the items in the ascending order of age of the author referred by referenced field "blogauthor".</p> <p>Example: <code>orderBy=taxonomies.shortName["loc"];taxonomies.categories.nodes.name["loc"] "Location", "emea", "china", "industry";taxonomies.categories.nodes.name["loc"] "Location", "americas", "brazil", "industry"</code> Returns all the items sorted in the ascending order of root category of "loc", sub-categories of category whose path is /Location/emea/china/industry, sub-categories of category whose path is /Location/americas/brazil/industry.</p> <p>Example: <code>orderBy=fields.<text_field_name>:asc</code> Returns all the items sorted in the ascending order using the user-defined text field.</p> <p>Example: <code>orderBy=fields.<json_field_name>.value.<attribute_name>:asc</code> Returns all the items sorted in the ascending order using an attribute in JSON data stored in JSON field (embedded content) on assets.</p> <p>Example: <code>orderBy=fields.<ref_field_name><text_field_name>:asc</code> Returns all the items sorted in the ascending order using two-level reference to the user-defined text field. The example uses the compatibility mode of the two-level search.</p> <p>Example: <code>orderBy=fields.<ref_field_name>.fields.<text_field_name>:asc&twolvl_v1_1=true</code></p>

Query Parameter	Type	Description
		Returns all the items sorted in the ascending order using two-level reference to the user-defined text field. The example uses the two-level mode of the two-level search.
q	string	<p>This parameter accepts a query expression condition that matches the field values. Many such query conditions can be joined using AND/OR operators and grouped with parentheses. The value of query condition follows the format of {fieldName} {operator} "{fieldValue}". In the case of a query across type, the field names are limited to standard fields (Id, type, name, description, typeCategory, slug, language, createDate, updatedAt, taxonomies). However in the case of a type-specific query, the field names are limited to standard fields and user-defined fields (except fields of largeText data type). The only values allowed in the operator are eq (Equals), co (Contains), sw (Startswith), ge (greater than or equals to), le (less than or equals to), gt (Greater than), lt (less than), mt (Matches), sm (Similar).</p> <p>Example: <code>https://{cecsdomain}/content/published/api/v1.1/items?q=(name eq "John")</code></p> <p>Example: <code>https://{cecsdomain}/content/published/api/v1.1/items?q=(type eq "Employee" AND name eq "John")</code></p> <p>Example: <code>https://{cecsdomain}/content/published/api/v1.1/items?q=(type eq "Employee" AND ((name eq "John" AND field.age ge "40") OR fields.weight gt "70"))</code></p> <p>Example: <code>https://{cecsdomain}/content/published/api/v1.1/items?q=(taxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489" OR (taxonomies.categories.name co "cat" AND taxonomies.categories.name co "red"))</code></p> <p>Example: <code>https://{cecsdomain}/content/published/api/v1.1/items?q=(taxonomies.categories.nodes.id eq "9E1A79EE600C4C4BB727FE3E39E95489" OR taxonomies.categories.nodes.name co "cars")</code></p>
scroll	boolean	<p>Specifying this parameter indicates that the scroll behavior is expected from the Search API. Scrolling is the recommended method for retrieving large result sets. Scrolling is not available when <code>returnMaster</code> is enabled. <code>hasMore</code> will always return false when scrolling is enabled. The offset parameter value, if specified, will be ignored on scroll requests. The limit parameter value will be interpreted in the context of scroll and be fixed for a scroll session. Subsequent changes to limit will be ignored for the scroll session. A limit that results in too large a response will result in a 413 (Payload Too Large) response status.</p> <p>Default Value: false</p>
scrollId	string	<p>This parameter is used to specify the scroll Id. <code>scrollTTL</code> and the original value of the <code>q</code> parameter are always required with requests that use a <code>scrollId</code> to get consistent results. Note that the requirement to always include the <code>q</code> parameter in subsequent scroll requests is currently not validated against previous requests and changes to the parameter within the same scroll session are ignored.</p>

Query Parameter	Type	Description
scrollTTL	integer(int32)	scrollTTL (in milliseconds - default and maximum value 30000 ms) specifies the period of inactivity allowed between the current and the next scroll request. All responses to search with scroll include a scrollId. Subsequent scroll requests must include the scrollId returned from the previous response since the scrollId returned could change across requests. Since all scroll requests are stateless, scrollTTL is always expected. The value of scrollTTL does not have to be the same across requests. The original search query (q) is also expected with each subsequent scroll request. An invalid or expired scrollId will result in a 400 (Bad Request) response status. Default Value: 30000
totalResults	boolean	Accepts a boolean value. Setting it to true displays the total results field in the response. The default value is false
twolvl_v1_1	string	This parameter does not need a value and when used it indicates the two level the Search API uses (i) <code>fields.refname.fieldname</code> to search on system fields such as name, description, etc of the referred type; and (ii) <code>fields.refname.fields.userfieldname</code> to search on user-defined fields of the referred type. When this parameter is not used, the two-level search behaves in compatibility mode and only supports searches like <code>fields.reftype.fieldname</code> . Furthermore, when <code>fieldname</code> is the same as a system-defined field name, the two-level search behaves the same as <code>fields.reftype.fields.fieldname</code> .

REST API for Content Management

The following table lists extra search request parameters accepted by the REST API for Content Management:

Query Parameter	Type	Description
channelToken	string	This parameter accepts channelToken of a channel and is used to control the returned results. The result will contain only items targeted to the channel that the specified channelToken belongs to. This can also be achieved by specifying the channels (standard field of an item) contains query condition (<code>channels co "{channelId}"</code>) as one of the query conditions in the <code>q</code> query parameter. This is an optional parameter and by default returns all the results.
default	string	Default search query expression which matches the values of the items across all fields.
expand	string	This parameter is used to allow users to get permissions on each matching item. Accepts permissions only.

Query Parameter	Type	Description
fields	string	<p>This parameter is used to control the returned fields in each item in the result. This parameter accepts a comma-separated list of field names or all. These fields will be returned for each items in the result. All the field names are case-sensitive, and users must provide the correct field name in the query. All the user-defined field names should be provided with prefix fields and followed by period (.). When <code>fields</code> is specified as all (case-insensitive), all the standard fields are returned in the case of a query across types and in the case of a type-specific query, all standard and user fields are returned. This parameter is optional in the query, and by default the result shows only standard field name and description. The standard fields, <code>Id</code> and <code>type</code>, are always returned irrespective of any field asked. Any incorrect or invalid field name given in the query will throw an error. In the context of a brace style cross-type query, type-specific fields may be specified using the syntax <code>name, {typename1:fields.userdefinedfieldname1,fields.userdefinedfieldname2}, {typename2:fields.userdefinedfieldname1}</code>. In the preceding example, all items of the type, <code>typename1</code>, will have fields - <code>name</code>, <code>userdefinedfieldname1</code>, and <code>userdefinedfieldname2</code>, while all items of the type, <code>typename2</code>, will have fields - <code>name</code>, <code>userdefinedfieldname1</code>. If the cross-type query does not resolve to types referenced in the typed fields clause(s), an error will be thrown.</p> <p>Example: This returns standard fields (<code>name</code>), user fields (<code>state</code>), and country of the type address in the search results.</p> <pre>https://{cecsdomain}/content/management/api/v1.1/items?q=type eq "Address"&fields=fields.state,fields.country</pre> <p>Example: This returns all the attributes for a specific type used in the search results.</p> <pre>https://{cecsdomain}/content/management/api/v1.1/items?q=type eq "Address"&fields=all</pre> <p>Example: This returns standard fields, <code>name</code> and <code>createdBy</code> in the search results for all items across all the types.</p> <pre>https://{cecsdomain}/content/management/api/v1.1/items?fields=name,createdBy</pre> <p>Example: This returns all the standard fields in the search results for all items across all the types.</p> <pre>https://{cecsdomain}/content/management/api/v1.1/items?fields=all</pre> <p>Default value: <code>name</code>, <code>description</code>, <code>repositoryId</code>, <code>slug</code>, <code>language</code>, <code>translatable</code>, <code>createdDate</code>, <code>updatedDate</code></p>
limit	integer(int32)	<p>This parameter accepts a non negative integer and is used to control the size of the result. If <code>offset+limit > 10000</code>, then the limit is treated as 10000-offset and gives results. The default value is 100.</p>
links	string	<p>This parameter accepts a comma-separated list of link names. By default, this parameter gives all the links applicable. Possible values are: <code>self</code>, <code>canonical</code>, <code>describedby</code>, <code>first</code>, <code>last</code>, <code>prev</code>, <code>next</code></p>

Query Parameter	Type	Description
offset	integer(int32)	This parameter accepts a non negative integer less than 10000 and is used to control the start index of the result. The default value is 0.

Query Parameter	Type	Description
orderBy	string	<p>The <code>orderBy</code> parameter is used to control the order (ascending/descending) of queried items.</p> <p>This parameter is optional in the query and by default the results are sorted in the descending order of <code>updatedAt</code> when the <code>default</code> parameter is empty. When the <code>default</code> parameter has value(s), the results are sorted by the relevance of tags of the items to the default values.</p> <p>This parameter accepts <code>featured()</code> or field name separated by a colon (:) for which the user wants to sort the results and sort order. Multiple sort orders are separated by semicolon (;).</p> <p>Format: <code>orderBy={fieldName1}:{asc/desc};{fieldName2}:{asc/desc}</code></p>

 **Note:**

asc stands for ascending and desc for descending. asc and desc are always in lower case.

In a type-specific query, field names can be either standard fields (`name`, `createdDate`, `updatedAt`) or user-defined fields (single-valued data types (`number`, `decimal`, `datetime`, `text`)). All the user-defined field names should be provided with prefix fields and followed by a period (.).

In the context of a cross-type search, this parameter can also have a typed section and takes the form `{typename1:fields.userdefinedfieldname1:asc|desc};{typename2:fields.userdefinedfieldname1:asc|desc}`. The `orderBy` parameter only supports one order by field per set of matching braces. While filtering on `suggestedTaxonomies`, the default sort order will be by relevance of the suggestion to the item.

The `orderBy` parameter also supports the following:

- Sorting by a two-level field. If there is a two-level sort field, then it can be the only sort field in the `orderBy` parameter.

Sorting assets using user-defined text field or an attribute in JSON data stored in JSON field (embedded content) on assets in a two-level search.

 **Note:**

The two-level sorting does not work on multiple value fields, so this does not work on multiple text fields. The two-level sorting works on any array attributes on JSON fields because JSON fields can only be a single value field.

- Sorting by taxonomy root categories and sub-categories of a root category. The additional sorting parameters are as follows:

Query Parameter	Type	Description
		<p>taxonomies.shortName["short name of taxonomy"]: {asc/desc}; taxonomies.categories.nodes.name["short name of taxonomy" "taxonomy name", "category1", "category2", "category3"] where category1, category2, category3, etc. are sub-categories of a given taxonomy, "taxonomy name". The sort order is optional, and by default it is descending. Any incorrect or invalid field name given in the query will be ignored. When featured() is given and the default parameter has value(s), the results are sorted by the relevance of tags of the items to the default values.</p> <ul style="list-style-type: none"> • Sorting by an attribute in JSON data stored in JSON field on assets. <p>Example: orderBy=name:asc Returns all the items in the ascending order of name.</p> <p>Example: orderBy=updateDate:asc Returns all the items in the ascending order of updateDate.</p> <p>Example: orderBy=fields.age:desc Returns all the items in the descending order of age.</p> <p>Example: orderBy=fields.age Returns all the items in the ascending order of age.</p> <p>Example: default=apples&orderBy=featured() Returns all the items sorted by the relevance of tags of the items to the apples</p> <p>Example: orderBy=fields.blogauthor.authorage Returns all the items in the ascending order of age of the author referred by referenced field, "blogauthor".</p> <p>Example: orderBy=fields.blogauthor.fields.authorage&twolvl_v1_1=true Returns all the items in the ascending order of age of the author referred by the referenced field, "blogauthor".</p> <p>Example: orderBy=taxonomies.shortName["loc"];taxonomies.categories.nodes.name["loc" "Location", "emea", "china", "industry"];taxonomies.categories.nodes.name["loc" "Location", "americas", "brazil", "industry"] Returns all the items in the ascending order of the root category of "loc", sub-categories of category whose path is /Location/emea/china/industry, sub-categories of category whose path is /Location/americas/brazil/industry.</p> <p>Example: orderBy=fields.<text_field_name>:asc Returns all the items sorted in the ascending order using the user-defined text field.</p> <p>Example: orderBy=fields.<json_field_name>.value.<attribute_name>:asc</p>

Query Parameter	Type	Description
		<p>Returns all the items sorted in the ascending order using an attribute in JSON data stored in JSON field (embedded content) on assets.</p> <p>Example:</p> <pre>orderBy=fields.<ref_field_name><text_field_name>:asc</pre> <p>Returns all the items sorted in the ascending order using two-level reference to the user-defined text field. The example uses the compatibility mode of the two-level search.</p> <p>Example:</p> <pre>orderBy=fields.<ref_field_name>.fields.<text_field_name>:asc&twolvl_vl_1=true</pre> <p>Returns all the items sorted in the ascending order using two-level reference to the user-defined text field. The example uses the two-level mode of the two-level search.</p>
q	string	<p>This parameter accepts a query expression condition that matches the field values. Many such query conditions can be joined using AND/OR operators and grouped with parentheses. The value of the query condition follows the format, {fieldName} {operator} "{fieldValue}". In case of a type-specific query, the field names are limited to standard fields and user-defined fields (except fields of the largeText data type). The only values allowed in the operator are eq (Equals), co (Contains), sw (Startswith), ge (Greater than or equals to), le (Less than or equals to), gt (Greater than), lt (Less than), mt (Matches), sm (Similar).</p> <p>Example: <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(name eq "John")</code></p> <p>Example: <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(type eq "Employee" AND name eq "John")</code></p> <p>Example: <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(type eq "Employee" AND ((name eq "John" AND field.age ge "40") OR fields.weight gt "70"))</code></p> <p>Example: <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(taxonomies.categories.id eq "9E1A79EE600C4C4BB727FE3E39E95489" OR (taxonomies.categories.name co "cat" AND taxonomies.categories.name co "red"))</code></p> <p>Example: <code>https://{cecsdomain}/content/management/api/v1.1/items?q=(taxonomies.categories.nodes.id eq "9E1A79EE600C4C4BB727FE3E39E95489" OR taxonomies.categories.nodes.name co "cars")</code></p>

Query Parameter	Type	Description
repositoryId	string	This parameter accepts the Id of a repository and is used to control the returned results. The result will contain only items belonging to the specified repository. This can also be achieved by specifying the repositoryId (standard field of an item) equals query condition (<code>repositoryId eq "{repositoryId}"</code>) as one of the query conditions in the <code>q</code> query parameter. This is an optional parameter and by default returns results from all the repositories.
scroll	boolean	Specifying this parameter indicates that the scroll behavior is expected from the Search API. Scrolling is the recommended method for retrieving large result sets. Scrolling is not available when <code>returnMaster</code> is enabled. The <code>hasMore</code> parameter will always return false when scrolling is enabled. The <code>offset</code> parameter value, if specified, will be ignored on scroll requests. The <code>limit</code> parameter value will be interpreted in the context of scroll and be fixed for a scroll session. Subsequent changes to limit will be ignored for the scroll session. A limit that results in too large a response will result in a 413 (Payload Too Large) response status. The default value is false.
scrollId	string	This parameter is used to specify the scroll Id. <code>scrollTTL</code> and the original value of the <code>q</code> parameter are always required with requests that use a <code>scrollId</code> to get consistent results. Note that the requirement to always include the <code>q</code> parameter in subsequent scroll requests is currently not validated against previous requests and changes to the parameter within the same scroll session are ignored.
scrollTTL		<code>scrollTTL</code> (in milliseconds - default and maximum value 30000 ms) specifies the period of inactivity allowed between the current and the next scroll request. All responses to search with scroll include a <code>scrollId</code> . Subsequent scroll requests must include the <code>scrollId</code> returned from the previous response since the <code>scrollId</code> returned could change across requests. Since all scroll requests are stateless, <code>scrollTTL</code> is always expected. The value of <code>scrollTTL</code> does not have to be the same across requests. The original search query (<code>q</code>) is also expected with each subsequent scroll request. An invalid or expired <code>scrollId</code> will result in a 400 (Bad Request) response status. The default value is 30000.
totalResults	boolean	This parameter accepts a boolean flag. If specified as true, then the returned result must include the total result count. The default value is false.
twoLv_v1_1	string	This parameter does not need a value. When it is used, it indicates the two levels that the Search API uses (i) <code>fields.refname.fieldname</code> to search on system fields such as name, description, etc of the referred type; and (ii) <code>fields.refname.fields.userfieldname</code> to search on user-defined fields of the referred type. When this parameter is not used, the two-level search behaves in compatibility mode and only supports searches like <code>fields.reftype.fieldname</code> . Furthermore, when <code>fieldname</code> is the same as a system-defined field name, the two-level search behaves the same as <code>fields.reftype.fields.fieldname</code> .

Two-Level Deep Search

By default a search query matches standard or user-defined data fields on the item itself. Data fields on a referenced item may also be searched so long as it is a direct reference. This effectively provides you a way to configure search query that delves two level deep into the data field hierarchy for an item, and therefore, to perform a two-level deep search matching asset standard or user-defined data fields. The two-level search provides the same query expression semantics as those provided while searching the fields of an item. For example:

A search query matching the user-defined field on a reference: `./.../api/v1.1/items?q=(type eq "Employee" AND fields.address.street eq "Main St")` Here, the address is the name of a reference type field on the type, Employee, which in turn has a field with the name, street.

 **Note:**

Searching recursively down an item's reference hierarchy past the first level reference is not supported. Sorting by second level fields is not supported either.

Two-level deep search supports two ways to configure a search query on standard and user-defined fields on a referenced item:

- **Compatibility mode**

This is a default mode. In the compatibility mode, a query expression supports field notations like `fields.ref-field.field-name`, where `ref-field` is the name of a reference or a media field and `field-name` is the name of a field on the referred asset type. Furthermore, when `field-name` is the same as the name of a standard field (For example, `Id`, `name`, or `description`), the two-level deep search behaves the same as `fields.ref-field.fields.field-name`, that is, it treats `field-name` as the name of a user-defined field. For example:

- Query `(type eq "parent-type" AND fields.ref-field.field-name eq "blah")` will match items of the type `parent-type` (parent item) that have the value *blah* assigned to the user-defined field `field-name` on the child item, which is set as a reference on the field `ref-field`.
- Query `(type eq "parent-type" AND fields.ref-field.name eq "blah")` will match items of type `parent-type` (parent item) that have the value *blah* assigned to the standard field, `name` on the child item, which is set as reference on the field, `ref-field`.

- **Two-level mode**

This mode is activated by setting the query parameter, `twolvl_v1_1 mode` to `true`. In this mode the two-level search query expression uses `fields.ref-field.field-name` to search on standard fields, such as `Id`, `name`, or `description` and `fields.ref-field.fields.field-name` to search on user-defined fields of the referenced asset type. For example:

- Query `(type eq "parent-type" AND fields.ref-field.field-name eq "blah")` will match items of the type `parent-type` (parent item) that have the value *blah*

assigned to a standard field, `field-name` on the child item, which is set as a reference on the field, `ref-field`.

- Query (`type eq "parent-type" AND fields.ref-field.fields.field-name eq "blah"`) will match items of the type, `parent-type` (parent item) that have the value *blah* assigned to a user-defined field, `field-name` on the child item, which is set as a reference on the field, `ref-field`.

Search JSON Data in JSON Fields

Search API allows you to return content items, or digital assets of custom digital asset types, based on search terms matching values assigned to attributes in JSON data that is stored in JSON fields on the item. In the Oracle Content Management web interface, the data field for storing JSON formatted data on an asset type is called Embedded Content. The API supports search terms that match base attributes, array elements, or attributes in a JSON object, including an array of objects. JSON data indexing currently imposes the following limitations on the total size of the indexed JSON field (< 2.5MB), JSON field key length (< 512B), and depth to match field in JSON object (10 levels).

Query syntax uses dot notation for field names to reference any key in the arbitrary JSON content, i.e. you can use `fields.<json_field_name>.value` to query custom text field or you can have `fields.<json_field_name>.value.key1.subkey1.subsubkey3`, etc. In a search condition, you can use all existing operators that are supported by Search API for regular custom fields, for example:

```
q=(type eq "Employee" AND fields.<json_field_name>.value.street eq
"Main St;)
q=(type eq "Employee" AND fields.<json_field_name>.value.age gt ;20")
q=(type eq "Employee" AND fields.<json_field_name>.value.proper`es eq
;office=HQ; AND fields.<json_field_name>.value.proper`es eq
;building=100;)
```

Similarly, for two-level deep search:

```
q=(type eq "Employee" AND
fields.<ref_field_name>.fields.<json_field_name>.value.street eq "Main
St;)
q=(type eq "Employee" AND
fields.<ref_field_name>.fields.<json_field_name>.value.age gt ;20")
q=(type eq "Employee" AND
fields.<ref_field_name>.fields.<json_field_name>.value.proper`es eq
;office=HQ; AND
fields.<ref_field_name>.fields.<json_field_name>.value.proper`es eq
;building=100;)
```

Use of wildcards in search conditions, such as `fields.<json_field_name>.value.*.subsubkey3` and such are not supported for performance reasons.

The Search API also allows you to sort returned search results by JSON Keys/subkeys, for example:

```
orderBy= fields.<json_field_name>.value.name:asc
orderBy= fields.<json_field_name>.value.name:desc
orderBy= fields.<ref_field_name>.fields.<json_field_name>.value.name:asc
orderBy= fields.<ref_field_name>.fields.<json_field_name>.value.name:desc
```

Search Across Types

Prior to the 21.6.1 release of Oracle Content Management (June 2021), specifying multiple asset types in a search query made it a global (untyped) query that only allows you to reference standard fields on an asset type in query conditions and in orderBy specifications. An asset type specific predicates like `fields.custom-filed-name`, which could not be used in the untyped query conditions, fields specification, or orderBy specification. The new cross-type search API and search query syntax associated with it removes these restrictions.

A search query on custom data fields across multiple asset types requires you to use a new syntax for building a search query expression. A query across multiple types is formulated by enclosing asset type predicates with curly brackets `{}`. The Search query is validated to comply with the following restrictions:

- Only one asset type may be specified within a matching pair of curly brackets. The `{}` pair defines a type scope.
- Nested asset type scope delimiters are not allowed, for example, `{...{...}...}` expression would fail validation.
- Curly brackets `{}` can be combined with square brackets `[]` when defining a typed expression for the `fields` or `orderBy` parameters.
- Predicates specified outside a type scope behave like they were specified using square brackets, that is, like a non-multi-type search query parameter.
- Queries do not support the `not equal (ne)` operator across different user-defined types.

Examples of search query on custom data fields across multiple asset types:

- `{type eq "t1" AND fields.ud1 eq "ud1"}`
ud1 has to be a field on the type, t1. Query will fail validation if that is not the case.
- `{type eq "t1" AND fields.ud1 eq "ud1"} AND name eq "John"`
Same as above with a standard field outside the type delimiter.
- `{type eq "t1" AND fields.ud1 eq "ud1"} OR {type eq "t2" AND fields.ud2 eq "ud2"}`
ud1 has to be a field on the type, t1 and ud2 has to be a field on the type, t2.
- `{type eq "t1" AND and fields.ud1 eq "ud1"} AND {type eq "t2" AND fields.ud2 eq "ud2"}`
Similar to the previous query and valid, but it will return zero results due to the AND operator.
- `{type eq "t1" OR type eq "t2"}`
Invalid query, since two types are specified within a type delimiter.
- `type eq "t1" OR type eq "t2"`
Un-scoped style search that resolves to an untyped query.
- `type eq "t1" OR type eq "t2" AND fields.ud1 eq "ud1"`

Un-scoped style search that resolves to untyped and will throw a validation error.

- `type eq "t1" AND fields.ud1 eq "ud1"`
Un-scoped style search that resolves to the type, t1. It is valid as long as ud1 is a field on t1.
- `{type eq "t2" AND fields.ud2 eq "ud2"} OR type eq "t1" AND fields.ud1 eq "ud1"`
Un-scoped style search that resolves to the type, t1, and scoped search that resolves to t2. Query is valid if ud1 is a field on t1 and ud2 is a field on t2.

The fields Parameter

The untyped way of configuring the `fields` parameter on request to return standard and user-defined data fields for a single type uses the form `Id, name, type, updatedDate, fields.ud1, fields.ud2`. The `fields` parameter can be typed as well and takes the form `{<typename1>:fields.<userdefinedfieldname>, [fields.<userdefinedfieldname>]}, {<typename2>:fields.<userdefinedfieldname>, [fields.<userdefinedfieldname>]}`.

If an untyped expression is used as the `fields` parameter value in a combination with a typed query, the `q` parameter has to resolve to a single un-scoped asset type (a type predicate specified outside `{}` delimiters), otherwise an exception will be thrown.

Some examples of the typed expression set as the `fields` parameter value:

- `fields=id,name,type,updatedDate,{t1:fields.ud1}, {t2:fields.ud2, fields.ud3}`
The typed way to request system-defined fields (`Id, name, type, updatedDate`) and user defined fields `ud1` for type `t1` and `ud2` and `ud3` for type `t2`. An exception will be thrown during validation if types and user defined fields don't match.
- `fields=id,name,type,updatedDate,{t1:fields.ud1}, {t2:fields.ud2}, fields.ud3`
The typed way to request system-defined fields (`id, name, type, updatedDate`) and user defined fields `ud1` for type `t1` and `ud2` for type `t2`. The user-defined field, `ud3`, must resolve to an unscoped type, otherwise an exception will be thrown during validation.

The orderBy Parameter

The `orderBy` parameter can also have a typed section, and it takes the form `{<typename1>:fields.<userdefinedfieldname>[:asc|desc]}; {<typename2>:fields.<userdefinedfieldname>[:asc|desc]}`. The `orderBy` parameter only supports one order by field per type scope.

Dynamic Count of Assets per Taxonomy Category

The dynamic assets per category category count is a named aggregation, where items matching a search query are further analyzed to answer the question: "How many items in the result set are assigned to a specific category?". It should be noted that the analyzer needs to reflect direct and indirect category assignments. For example, if an item is assigned to a leaf level category, the item count should propagate to all parent level categories. If the same item is assigned to multiple leaf level categories, it should be counted as one item on the common parent category (and all its parent categories).

In other words, duplicate assignments should not be counted. Consider the following example:

Given the following category tree in your taxonomy:	Item count per category should account for a product item D500 assigned to both DSLR and Nikon as follows:
Cameras	Cameras (1)
Type	Type (1)
DSLR	DSLR (1)
Brand	Brand (1)
Nikon	Nikon (1)

The same result is expected if the item, D500, is also directly assigned to the Cameras category.

If you use taxonomy to allow end-users of a website or application to browse categorized items (for example, in a product catalog on your website), you can use the item count per category to dynamically filter out and hide categories which have no items assigned, while leaving categories that do have assigned items plus, you can display the actual count of assets on each category. Here is how your API client can do that by going through the following repeating cycle to narrow down category listing:

- Start with an initial filter query
- End-user adds a filter term or selects a category
- Run modified filter query plus, an aggregation query(s) as required
- Narrow down the category listing based on the returned aggregated counts
- Cycle repeats from #2 as end-user continues with filtering items

The AGGS Query Parameter

To execute a search query that returns an aggregated item count per category, you need to use the `aggs` query parameters that takes JSON array as a value. Each element in this array represents an aggregation which is defined by required and optional attributes described in the table below. The `/items` resource supports the `aggs` parameter in both the REST API for Content Delivery and REST API for Content Management.

Query Parameter	Type	Aggregation Attributes
<code>aggs</code>	JSON array	<ul style="list-style-type: none"> • name (required): <code>item_count_per_category</code> • field (optional): Takes either <code>Id</code>, or <code>apiname</code> to specify if <code>Id</code> or <code>apiName</code> is returned for a category. By default, the category <code>Id</code> is returned. • size (optional): Takes a number in the <code>[1...1000]</code> range. By default 1000 is used.

Some examples of search queries with aggregations:

- A query with a single aggregation

```
.../api/v1.1/items?
fields=name&aggs={"name":"item_count_per_category"}&q=(type eq
"ContentType1" AND fields.simple_text_value mt "brown")
```

- A query with a single aggregation and with the query parameter, `limit=0` will return no search results. Aggregated counts will be returned from the Elasticsearch aggregation cache

```
.../api/v1.1/items?
fields=name&aggs={"name":"item_count_per_category"}&q=(type eq
"ContentType1" AND fields.simple_text_value mt "brown")&limit=0
```

- A query with two aggregations

```
.../api/v1.1/items?
fields=name&aggs=[{"name":"item_count_per_category"},
{"name":"item_count_per_category","field":"apiname"}]&q=(type eq
"ContentType1" AND fields.simple_text_value mt "brown")
```

The aggregation section in the response to the search request for the above sample query with two aggregations is shown below:

```
"aggregationResults": [
{
  "itemCountPerCategory": [
    {
      "categoryId": "65EAC164681C47D68851583237381001",
      "itemCount": 3
    },
    {
      "categoryId": "C1E24A8BA3754F5AB17AEED3F020A898",
      "itemCount": 3
    },
    {
      "categoryId": "AF8D57438E3645ABBB137812FE830C34",
      "itemCount": 2
    }
  ],
  "name": "item_count_per_category"
},
{
  "itemCountPerCategory": [
    {
      "categoryApiName": "tax-cat1",
      "itemCount": 3
    },
    {
      "categoryApiName": "tax-cat2",
      "itemCount": 3
    },
    {
      "categoryApiName": "tax-cat3",
      "itemCount": 2
    }
  ],
  "name": "item_count_per_category"
}
]
```

Aggregation Cache

The aggregated asset count per category API relies on the underlying Elasticsearch requests cache, which is enabled by default on the Oracle Content Management search. By default, the request cache will only cache the results of search requests when the query parameter `limit=0`, so that it does not cache matching items, but it will cache the `hits.total`, `aggregations` and `suggestions`. This allows frequently used (and potentially heavy) search requests to return results almost instantly. The requests cache keeps the same near real-time promise as uncached search. Elasticsearch invalidates cached results automatically whenever the shard refreshes, but only if the data in the shard has actually changed. In other words, Elasticsearch always get the same results from the cache as it would for an uncached search request.

Using Dynamic Asset Counts per Category API

The dynamic item per category aggregation API can be utilized to display counts as required by the common e-commerce product selectors / product catalogs. This section describes the approach for a typical e-commerce use case as well as more general use cases. The following terms are used to describe the rules for computing the aggregate asset count per category:

- **User query:** A user-specified search query expression of arbitrary complexity conforming to Oracle Content Management Search API query syntax requirements. This is a free form search query on whatever an end-user entered in the input, such as (name eq "car").
- **Category node query:** A form of Oracle Content Management Search API query expressed as `'(taxonomies.categories.nodes.id eq "<id_1>" OR taxonomies.categories.nodes.id eq "<id_2>" OR ... OR taxonomies.categories.nodes.id eq "<id_N>")'`, where `id_x` is the Id of one of the user-selected categories under a particular taxonomy.
- **Search query:** One or more search queries intended to fetch results for display to end-users. It can be a user query, one or more category node queries, or a combination of those.
- **Aggregation query:** A variant of user query, generated to compute the aggregated item count for a specific search facet.

Computing Aggregate Asset Counts

No user-selected categories: With no user-selected categories you can use a user query to get matching items and compute aggregated counts:

- **Search query:** `q=<user_query>`
- **Aggregation query (all taxonomies):** `q=<user_query>`

User selects one or more categories from one taxonomy - taxonomy-1:

- **Search query:** `q=<user_query> AND <category_node_query_1>`
- **Aggregation query(taxonomy-1):** `q=<user_query>`
- **Aggregation query (all other taxonomies):** `q=<search_query>`

User selects categories from two taxonomy - taxonomy-1 and taxonomy-2:

- **Search query:** q=<user_query> AND <category_node_query_1> AND <category_node_query_2>
- **Aggregation query (taxonomy-1):** q=<user_query> AND <category_node_query_2>
- **Aggregation query (taxonomy-2):** q=<user_query> AND <category_node_query_1>
- **Aggregation query (all other taxonomies):** q=<search_query>

User selects categories from three taxonomy - taxonomy-1, taxonomy-2 and taxonomy-3:

- **Search query:** q=<user_query> AND <category_node_query_1> AND <category_node_query_2> AND <category_node_query_3>
- **Aggregation query (taxonomy-1):** q=<user_query> AND <category_node_query_2> AND <category_node_query_3>
- **Aggregation query (taxonomy-2):** q=<user_query> AND <category_node_query_1> AND <category_node_query_3>
- **Aggregation query (taxonomy-3):** q=<user_query> AND <category_node_query_1> AND <category_node_query_2>

A general rule, where user selects categories from N taxonomies:

- **Search query:** q=<user_query> AND <category_node_query_1> AND <category_node_query_2> ... AND <category_node_query_N>
- **Aggregation query (taxonomy-1):** q=<user_query> AND <category_node_query_2> ... AND <category_node_query_N>
- **Aggregation query (taxonomy-2):** q=<user_query> AND <category_node_query_1> AND <category_node_query_3> ... AND <category_node_query_N>
- **Aggregation query (taxonomy-N):** q=<user_query> AND <category_node_query_1> AND <category_node_query_2> ... AND <category_node_query_N-1>
- **Aggregation query (all other taxonomies):** q=<search_query>

Total Search Cost

One scored/ranked query to get search results plus aggregate counts for categories under taxonomies with no user-selected categories. N filter queries to compute aggregate counts for N taxonomies with selected categories.

An e-commerce Use Case

The table below lists three taxonomies that are used in this example. It also includes a list of items in repository and item categorization with taxonomies that you can refer to for this example:

Taxonomies	Items and Item Categorization	Item Count per Category
Electronics (ELE) ELE-laptops ELE-keyboards	All items below have the same content type <i>ContentType2:</i> <ul style="list-style-type: none"> HP Elite Dragonfly (added to: ELE-laptops, MAN-hp, LOC-sheraton-mall, LOC-stoneridge-mall) 	Electronics (ELE) ELE-laptops (5) ELE-keyboards (4)
Manufacturer (MAN) MAN-hp MAN-razer	<ul style="list-style-type: none"> HP Elite Folio (added to: ELE-laptops, MAN-hp, LOC-sheraton-mall, LOC-stoneridge-mall) HP Elite X2 (added to: ELE-laptops, MAN-hp, LOC-sheraton-mall, LOC-stoneridge-mall) HP Pavillion Keyboard (added to: ELE-keyboards, MAN-hp, LOC-sheraton-mall, LOC-stoneridge-mall) 	Manufacturer (MAN) MAN-hp (5) MAN-razer (4)
Location (LOC) LOC-sheraton-mall LOC-stoneridge-mall	<ul style="list-style-type: none"> HP Omen Keyboard (added to: ELE-keyboards, MAN-hp, LOC-sheraton-mall, LOC-stoneridge-mall) Razer Blade Stealth(added to: ELE-laptops, MAN-razer, LOC-sheraton-mall) Razer Blade Pro(added to: ELE-laptops, MAN-razer, LOC-sheraton-mall) Razer BlackWidow Keyboard (added to: ELE-laptops, MAN-razer, LOC-sheraton-mall) Razer Huntsman Keyboard (added to: ELE-laptops, MAN-razer, LOC-sheraton-mall) 	Location (LOC) LOC-sheraton-mall (9) LOC-stoneridge-mall (5)

Let's use a simple search query to match all assets for No Categories is Selected to match all assets of the type, Content Type2, in a given repository. To make queries and API response to a search query request easier to read, all categories in the sample taxonomy have the same API name (apiName) as the category name. Therefore, all search query expressions and aggregations below use the apiName. You can use category Id instead.

No Categories is Selected

If end-users didn't select any category yet, you can obtain aggregated asset counts per category using the search query alone.

- Search query:**

```
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2")&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}];
```
- Aggregation query (all taxonomies):**

```
q=<search_query>
```

Search query response:

```
{
  "hasMore": false,
  "offset": 0,
  "count": 9,
  "limit": 9,
  "items": [
    {
      "name": "HP Omen Keyboard",
      "links": [],

```

```
    "id": "COREFE86EA24BAE84D4E87B8A1F992163D6B",
    "type": "ContentType2"
  },
  {
    "name": "Razer BlackWidow Keyboard",
    "links": [],
    "id": "COREF8CEBA1264C04A1A87954C2A4A33D2DB",
    "type": "ContentType2"
  },
  {
    "name": "Razer Huntsman Keyboard",
    "links": [],
    "id": "CORE07C7E1108BFB4A5EAF1761F46251FECD",
    "type": "ContentType2"
  },
  {
    "name": "HP Pavillion Keyboard",
    "links": [],
    "id": "COREAB4987C5FB664007B3B6B4D93CFAC90F",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite X2",
    "links": [],
    "id": "COREF911737800FE406C92183E182C7228CC",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite Folio",
    "links": [],
    "id": "CORE18B3AFABE2814E75AB54CBBF80B3A509",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite Dragonfly",
    "links": [],
    "id": "COREAED2F7C50B404D6FB6547CCE69DD3492",
    "type": "ContentType2"
  },
  {
    "name": "Razer Blade Pro",
    "links": [],
    "id": "CORE66D9CFFF1C2945A7857C6C0B5E4B21BB",
    "type": "ContentType2"
  },
  {
    "name": "Razer Blade Stealth",
    "links": [],
    "id": "CORED009A429B9D04080AE19FF42CE8CC758",
    "type": "ContentType2"
  }
],
"links": [],
"aggregationResults": [
  {
```

```

    "itemCountPerCategory": [
      {
        "categoryApiName": "loc-sheraton-mall",
        "itemCount": 9
      }, {
        "categoryApiName": "ele-laptops",
        "itemCount": 5 },
      {
        "categoryApiName": "loc-stoneridge-mall",
        "itemCount": 5
      }, {
        "categoryApiName": "man-hp",
        "itemCount": 5 },
      {
        "categoryApiName": "ele-keyboards",
        "itemCount": 4
      }, {
        "categoryApiName": "man-razer",
        "itemCount": 4 }
    ],
    "name": "item_count_per_category"
  }
]
}

```

End-user selected "ELE-laptops"

With one category selected, let's use a search query to get results and obtain aggregated counts for taxonomies with no selected categories. Then we will use a separate aggregate query to compute aggregated counts for the taxonomy, Electronics. The aggregate query can use the parameter `limit=0` to return no items, as we are only interested in aggregated counts.

- **Search query:**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "ele-
laptops"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","f
ield":"apiname"}]

```

Search query response:

```

{
  "hasMore": false,
  "offset": 0,
  "count": 5,
  "limit": 5,
  "items": [
    {
      "name": "HP Elite X2",
      "links": [],
      "id": "COREF911737800FE406C92183E182C7228CC",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite Folio",
      "links": [],
      "id": "CORE18B3AFABE2814E75AB54CBBF80B3A509",

```

```

        "type": "ContentType2"
    },
    {
        "name": "HP Elite Dragonfly",
        "links": [],
        "id": "COREAED2F7C50B404D6FB6547CCE69DD3492",
        "type": "ContentType2"
    },
    {
        "name": "Razer Blade Pro",
        "links": [],
        "id": "CORE66D9CFFF1C2945A7857C6C0B5E4B21BB",
        "type": "ContentType2"
    },
    {
        "name": "Razer Blade Stealth",
        "links": [],
        "id": "CORED009A429B9D04080AE19FF42CE8CC758",
        "type": "ContentType2"
    }
],
"links": [],
"aggregationResults": [
    {
        "itemCountPerCategory": [
            {
                "categoryApiName": "ele-laptops",
                "itemCount": 5
            },
            {
                "categoryApiName": "loc-sheraton-mall",
                "itemCount": 5
            },
            {
                "categoryApiName": "loc-stoneridge-mall",
                "itemCount": 3
            },
            {
                "categoryApiName": "man-hp",
                "itemCount": 3
            },
            {
                "categoryApiName": "man-razer",
                "itemCount": 2
            }
        ]
    },
    {
        "name": "item_count_per_category"
    }
]
}

```

- **Aggregation query (for electronics):**

```

q=(repositoryId eq "<repo_ID>" AND type eq
"ContentType2")&fields=name&limit=0&links=none
&aggs[{"name":"item_count_per_category","field":"apiname"}]

```

Aggregation query response:

```

{
  "hasMore": false,
  "offset": 0,
  "count": 0,
  "limit": 0,
  "items": [],
  "links": [],
  "aggregationResults": [
    {
      "itemCountPerCategory": [
        {
          "categoryApiName": "loc-sheraton-mall",
          "itemCount": 9
        },
        {
          "categoryApiName": "ele-laptops",
          "itemCount": 5
        },
        {
          "categoryApiName": "loc-stoneridge-mall",
          "itemCount": 5
        },
        {
          "categoryApiName": "man-hp",
          "itemCount": 5
        },
        {
          "categoryApiName": "ele-keyboards",
          "itemCount": 4
        },
        {
          "categoryApiName": "man-razer",
          "itemCount": 4
        }
      ],
      "name": "item_count_per_category"
    }
  ]
}

```

End-user selected "ELE-laptops" plus "MAN-hp"• **Search query:**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "ele-laptops") AND
(taxonomies.categories.nodes.apiName eq "man-
hp"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field"
:"apiname"}]

```

Search query response

```

{
  "hasMore": false,
  "offset": 0,

```

```

"count": 3,
"limit": 3,
"items": [
  {
    "name": "HP Elite Dragonfly",
    "links": [],
    "id": "CORE02B5199C0C154F9AB29C42F829A47F21",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite Folio",
    "links": [],
    "id": "COREAAC11256104F4D73AAF599230621A789",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite X2",
    "links": [],
    "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
    "type": "ContentType2"
  }
],
"links": [],
"aggregationResults": [
  {
    "itemCountPerCategory": [
      {
        "categoryApiName": "ele-laptops",
        "itemCount": 3
      },
      {
        "categoryApiName": "loc-sheraton-mall",
        "itemCount": 3
      },
      {
        "categoryApiName": "loc-stoneridge-mall",
        "itemCount": 3
      },
      {
        "categoryApiName": "man-hp",
        "itemCount": 3
      }
    ],
    "name": "item_count_per_category"
  }
]
}

```

- **Aggregation query (for electronics):**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "man-
hp"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_cat
egory","field":"apiname"}]

```

Aggregation query response

```
{
  "hasMore": false,
  "offset": 0,
  "count": 5,
  "limit": 5,
  "items": [
    {
      "name": "HP Elite Dragonfly",
      "links": [],
      "id": "CORE02B5199C0C154F9AB29C42F829A47F21",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite Folio",
      "links": [],
      "id": "COREAAC11256104F4D73AAF599230621A789",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite X2",
      "links": [],
      "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
      "type": "ContentType2"
    },
    {
      "name": "HP Pavillion Keyboard",
      "links": [],
      "id": "COREF0AED9516AED4DC6A0692A696170FD57",
      "type": "ContentType2"
    },
    {
      "name": "HP Omen Keyboard",
      "links": [],
      "id": "CORE816259C0386D4704AB006B9F40D9F457",
      "type": "ContentType2"
    }
  ],
  "links": [],
  "aggregationResults": [
    {
      "itemCountPerCategory": [
        {
          "categoryApiName": "loc-sheraton-mall",
          "itemCount": 5
        },
        {
          "categoryApiName": "loc-stoneridge-mall",
          "itemCount": 5
        },
        {
          "categoryApiName": "man-hp",
          "itemCount": 5
        }
      ]
    }
  ]
}
```



```

        {
          "categoryApiName": "ele-laptops",
          "itemCount": 3
        },
        {
          "categoryApiName": "ele-keyboards",
          "itemCount": 2
        }
      ],
      "name": "item_count_per_category"
    }
  ]
}

```

- **Aggregation query (for manufacturer):**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "ele-
laptops"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_pe
r_category","field":"apiname"}]

```

Aggregation query response

```

{
  "hasMore": false,
  "offset": 0,
  "count": 5,
  "limit": 5,
  "items": [
    {
      "name": "HP Elite Dragonfly",
      "links": [],
      "id": "CORE02B5199C0C154F9AB29C42F829A47F21",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite Folio",
      "links": [],
      "id": "COREAAC11256104F4D73AAF599230621A789",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite X2",
      "links": [],
      "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
      "type": "ContentType2"
    },
    {
      "name": "Razer Blade Stealth",
      "links": [],
      "id": "COREF49CB87945574CC59B2ACAE38A46D529",
      "type": "ContentType2"
    },
    {
      "name": "Razer Blade Pro",
      "links": [],

```

```

        "id": "CORE51A676D1B2FF4DB5B975806AFAF69284",
        "type": "ContentType2"
    }
],
"links": [],
"aggregationResults": [
    {
        "itemCountPerCategory": [
            {
                "categoryApiName": "ele-laptops",
                "itemCount": 5
            },
            {
                "categoryApiName": "loc-sheraton-mall",
                "itemCount": 5
            },
            {
                "categoryApiName": "loc-stoneridge-mall",
                "itemCount": 3
            },
            {
                "categoryApiName": "man-hp",
                "itemCount": 3
            },
            {
                "categoryApiName": "man-razer",
                "itemCount": 2
            }
        ],
        "name": "item_count_per_category"
    }
]
}

```

End-user selected "ELE-laptops" plus "MAN-hp" and "LOC-stonebrifge-mall"

- **Search query:**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "ele-laptops") AND
(taxonomies.categories.nodes.apiName eq "man-hp") AND
(taxonomies.categories.nodes.apiName eq "loc-stoneridge-
mall"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","fiel
d":"apiname"}]

```

Search query response

```

{
  "hasMore": false,
  "offset": 0,
  "count": 3,
  "limit": 3,
  "items": [
    {
      "name": "HP Elite Dragonfly",
      "links": [],
      "id": "CORE02B5199C0C154F9AB29C42F829A47F21",

```

```

        "type": "ContentType2"
      },
      {
        "name": "HP Elite Folio",
        "links": [],
        "id": "COREAAC11256104F4D73AAF599230621A789",
        "type": "ContentType2"
      },
      {
        "name": "HP Elite X2",
        "links": [],
        "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
        "type": "ContentType2"
      }
    ],
    "links": [],
    "aggregationResults": [
      {
        "itemCountPerCategory": [
          {
            "categoryApiName": "ele-laptops",
            "itemCount": 3
          },
          {
            "categoryApiName": "loc-sheraton-mall",
            "itemCount": 3
          },
          {
            "categoryApiName": "loc-stoneridge-mall",
            "itemCount": 3
          },
          {
            "categoryApiName": "man-hp",
            "itemCount": 3
          }
        ],
        "name": "item_count_per_category"
      }
    ]
  }
}

```

- **Aggregation query (for electronics):**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "man-hp") AND
(taxonomies.categories.nodes.apiName eq "loc-stoneridge-
mall"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_c
ategory","field":"apiname"}]

```

Aggregation query response

```

{
  "hasMore": false,
  "offset": 0,
  "count": 5,
  "limit": 5,

```

```
"items": [
  {
    "name": "HP Elite Dragonfly",
    "links": [],
    "id": "CORE02B5199C0C154F9AB29C42F829A47F21",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite Folio",
    "links": [],
    "id": "COREAAC11256104F4D73AAF599230621A789",
    "type": "ContentType2"
  },
  {
    "name": "HP Elite X2",
    "links": [],
    "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
    "type": "ContentType2"
  },
  {
    "name": "HP Pavillion Keyboard",
    "links": [],
    "id": "COREF0AED9516AED4DC6A0692A696170FD57",
    "type": "ContentType2"
  },
  {
    "name": "HP Omen Keyboard",
    "links": [],
    "id": "CORE816259C0386D4704AB006B9F40D9F457",
    "type": "ContentType2"
  }
],
"links": [],
"aggregationResults": [
  {
    "itemCountPerCategory": [
      {
        "categoryApiName": "loc-sheraton-mall",
        "itemCount": 5
      },
      {
        "categoryApiName": "loc-stoneridge-mall",
        "itemCount": 5
      },
      {
        "categoryApiName": "man-hp",
        "itemCount": 5
      },
      {
        "categoryApiName": "ele-laptops",
        "itemCount": 3
      },
      {
        "categoryApiName": "ele-keyboards",
        "itemCount": 2
      }
    ]
  }
]
```

```

    }
  ],
  "name": "item_count_per_category"
}
]
}

```

- **Aggregation query (for manufacturer):**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "ele-laptops") AND
(taxonomies.categories.nodes.apiName eq "loc-stoneridge-
mall"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_c
ategory","field":"apiname"}]

```

Aggregation query response

```

{
  "hasMore": false,
  "offset": 0,
  "count": 3,
  "limit": 3,
  "items": [
    {
      "name": "HP Elite Dragonfly",
      "links": [],
      "id": "CORE02B5199C0C154F9AB29C42F829A47F21",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite Folio",
      "links": [],
      "id": "COREAAC11256104F4D73AAF599230621A789",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite X2",
      "links": [],
      "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
      "type": "ContentType2"
    }
  ],
  "links": [],
  "aggregationResults": [
    {
      "itemCountPerCategory": [
        {
          "categoryApiName": "ele-laptops",
          "itemCount": 3
        },
        {
          "categoryApiName": "loc-sheraton-mall",
          "itemCount": 3
        },
        {
          "categoryApiName": "loc-stoneridge-mall",

```

```

        "itemCount": 3
      },
      {
        "categoryApiName": "man-hp",
        "itemCount": 3
      }
    ],
    "name": "item_count_per_category"
  }
]
}

```

- **Aggregation query (for location):**

```

q=(repositoryId eq "<repo_ID>" AND type eq "ContentType2" AND
(taxonomies.categories.nodes.apiName eq "ele-laptops") AND
(taxonomies.categories.nodes.apiName eq "man-
hp"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category"
,"field":"apiname"}]

```

Aggregation query response

```

{
  "hasMore": false,
  "offset": 0,
  "count": 3,
  "limit": 3,
  "items": [
    {
      "name": "HP Elite Dragonfly",
      "links": [],
      "id": "CORE02B5199C0C154F9AB29C42F829A47F21",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite Folio",
      "links": [],
      "id": "COREAAC11256104F4D73AAF599230621A789",
      "type": "ContentType2"
    },
    {
      "name": "HP Elite X2",
      "links": [],
      "id": "COREDCF70DB8B3CA4479876D759A3B7A39AC",
      "type": "ContentType2"
    }
  ],
  "links": [],
  "aggregationResults": [
    {
      "itemCountPerCategory": [
        {
          "categoryApiName": "ele-laptops",
          "itemCount": 3
        },
        {

```

```
        "categoryApiName": "loc-sheraton-mall",  
        "itemCount": 3  
    },  
    {  
        "categoryApiName": "loc-stoneridge-mall",  
        "itemCount": 3  
    },  
    {  
        "categoryApiName": "man-hp",  
        "itemCount": 3  
    }  
],  
"name": "item_count_per_category"  
}  
]  
}
```

A General Use Case

The table below includes the three taxonomies that are used in the example. It also includes a list of items in the repository and the item categorization with taxonomies that you can refer to:

Taxonomies	Items and Item Categorization	Item Count per Category
	All items below have the same content type, ContentType1:	
TA1		TA1
TA1-cat_0	• <i>Item1</i> (added to: TA1-cat_0_1_1_1)	TA1-cat_0 (2)
TA1-cat_0_1	• <i>Item2</i> (added to: TA1-cat_0_1_1, TA1-cat_0_2_1, TA2-cat_0_1_1, TA3-cat_0_1_1)	TA1-cat_0_1 (2)
TA1-cat_0_1_1	• <i>Item3</i> (added to: TA3-cat_0_1_1_1)	TA1-cat_0_1_1 (2)
TA1-cat_0_1_1_1	• <i>Item4</i> (added to: TA2-cat_0_1, TA3-cat_0_1)	TA1-cat_0_1_1_1 (1)
TA1-cat_0_2	• <i>Item5</i> (added to: TA3-cat_0_1)	TA1-cat_0_2 (1)
TA1-cat_0_2_1		TA1-cat_0_2_1 (1)
TA1-cat_0_2_1_1		TA1-cat_0_2_1_1 (0)
TA2		TA2
TA2-cat_0		TA2-cat_0 (2)
TA2-cat_0_1		TA2-cat_0_1 (2)
TA2-cat_0_1_1		TA2-cat_0_1_1 (1)
TA2-cat_0_1_1_1		TA2-cat_0_1_1_1 (0)
TA3		TA3
TA3-cat_0		TA3-cat_0 (4)
TA3-cat_0_1		TA3-cat_0_1 (4)
TA3-cat_0_1_1		TA3-cat_0_1_1 (2)
TA3-cat_0_1_1_1		TA3-cat_0_1_1_1 (1)

Just like in [the e-commerce use case](#), we will use a simple search query to match all assets for No Categories is Selected to match all assets of the type, Content Type1 in a given repository. To make queries and API response to a search query request easier to read, all categories in the sample taxonomy have the same API name (apiName) as the category name. Therefore, all search query expressions and aggregations below use the apiName. You can use the category Id instead.

No Categories is Selected

With no end-user selection made, we can obtain aggregate asset counts per category using a search query alone.

- **Search query:**

```
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1")&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]
```


- **Aggergate query:**
q=<search_query>

End-user deleted TA1-cat_0_1_1 plus TA1-cat_0_2_1

- **Search query:**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta1-cat_0_1_1" OR taxonomies.categories.nodes.apiName eq "ta1-cat_0_2_1"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]
- **Aggregation query (for TA1):**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1"&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]
- **Aggregation query (taxonomies with no category selected):**
q=<search_query>

End-user selected TA2-cat_0_1_1

- **Search query:**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1_1"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]
- **Aggregation query (for TA2):**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1"&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]
- **Aggregation query (taxonomies with no category selected):**
q=<search_query>

End-user selected TA3-cat_0_1_1

- **Search query:**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta3-cat_0_1_1"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]
- **Aggregation query (for TA3):**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1"&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]
- **Aggregation query (taxonomies with no category selected):**
q=<search_query>

End-user selected TA1-cat_0_1_1, TA1-cat_0_2_1 plus TA2-cat_0_1_1

- **Search query:**
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta1-cat_0_1_1" OR taxonomies.categories.nodes.apiName eq "ta1-cat_0_2_1") AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1_1"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]

- **Aggregation query (for TA1):**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query (for TA2):**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta1-cat_0_1_1" OR taxonomies.categories.nodes.apiName eq "ta1-cat_0_2_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query (taxonomies with no category selected):**
`q=<search_query>`

End-user selected TA2-cat_0_1 plus TA3-cat_0_1

- **Search query:**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1") AND (taxonomies.categories.nodes.apiName eq "ta3-cat_0_1"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query (for TA2):**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta3-cat_0_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query (for TA3):**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query (taxonomies with no selected categories):**
`q=<search_query>`

End-user selected TA1-cat_0_1_1, TA1-cat_0_2_1, TA2-cat_0_1_1 plus TA3-cat_0_1_1

- **Search query:**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta1-cat_0_1_1" OR taxonomies.categories.nodes.apiName eq "ta1-cat_0_2_1") AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1_1") AND (taxonomies.categories.nodes.apiName eq "ta3-cat_0_1_1"))&fields=name&links=none&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query(for TA1):**
`q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND (taxonomies.categories.nodes.apiName eq "ta2-cat_0_1_1") AND (taxonomies.categories.nodes.apiName eq "ta3-cat_0_1_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_per_category","field":"apiname"}]`
- **Aggregation query(for TA2):**

```
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND
(taxonomies.categories.nodes.apiName eq "ta1-cat_0_1_1" OR
taxonomies.categories.nodes.apiName eq "ta1-cat_0_2_1") AND
(taxonomies.categories.nodes.apiName eq "ta3-
cat_0_1_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_
per_category","field":"apiname"}]
```

- **Aggregation query(for TA3):**

```
q=(repositoryId eq "<repo_ID>" AND type eq "ContentType1" AND
(taxonomies.categories.nodes.apiName eq "ta1-cat_0_1_1" OR
taxonomies.categories.nodes.apiName eq "ta1-cat_0_2_1") AND
(taxonomies.categories.nodes.apiName eq "ta2-
cat_0_1_1"))&fields=name&links=none&limit=0&aggs=[{"name":"item_count_
per_category","field":"apiname"}]
```

- **Aggregation query (taxonomies with no selected categories):**

```
q=<search_query>
```

Scroll API

Scrolling or a scroll request is the recommended way of retrieving enormous search result sets. It is invoked by adding the boolean parameter, `scroll=true`, to a search request. The result of the first scroll search request, if successful, returns a `scrollId` in the response. All subsequent scroll requests of a first scroll request must supply the returned `scrollId` as a parameter, `scrollId=<scrollId value>`. The `scrollId` may change across subsequent scroll requests, so clients should always send the `scrollId` returned by the previous scroll response in a session. The `q` parameter specified with the first scroll request must also be specified with each subsequent scroll request. All results matched by a scroll search will have been retrieved when a scroll request returns 0 results or no `scrollId` is set on the response.

The number of results returned per page of a scroll search request is determined by the `limit` parameter. The `limit` parameter specified for the first scroll request will stay in effect for the life of the scroll session. As in other cases, a limit value that results in too large of a search response will result in a HTTP status code 413.

Scrolling is an expensive operation for the search server, so scroll sessions are aggressively cleaned up. If more than `scrollTTL` milliseconds elapse between subsequent scroll requests, the scroll session is cleaned up and any further usage of the session `scrollId` will result in an error. The maximum (and default) value for `scrollTTL` is 30000 ms (30s). It is recommended to have shorter `scrollTTL` values if feasible.



Note:

The `hasMore` parameter will always return false for scroll searches. Scroll cannot be used when the `returnMaster` parameter is set to true.

Example of a Scroll API Session

First scroll request: `/api/v1.1/items?q=(type eq "Employee" AND fields.role eq "Senior Developer")&scroll=true&scrollTTL=5000`

- Truncated first scroll response with `scrollId`: `scrollId=abcdefghijkl`

Subsequent scroll request: `/api/v1.1/items?q=(type eq "Employee" AND fields.role eq "Senior Developer")&scrollId=abcdefghijkl&scrollTTL=3000` (note `scrollTTL` can change)

- Truncated subsequent scroll response with `scrollId`: `scrollId=pqrstuvwxyz`

Subsequent scroll request: `/api/v1.1/items?q=(type eq "Employee" AND fields.role eq "Senior Developer")&scrollId=pqrstuvwxyz&scrollTTL=3000`

- Truncated final scroll response: `count=0`

Custom Ranking Policies

The REST API for Content Delivery, by default, applies a built-in ranking policy described below to calculate the relevance score for items matching a search query. Oracle Content Management allows you to override a built-in policy and control the relevance score of items matching a search query in the REST API for Content Delivery by defining a custom ranking policy, which contains a collection of ranking methods with configurable properties that relate to:

- Searchable standard and custom fields on content items or attributes on digital assets
- Indexed items to ranking method matching condition
- Impact on the item relevance score impact: Score increase or decay.
- Decay function

The search in the REST API for Content Management, by default, applies a built-in ranking policy to calculate the relevance score for items matching a search query. The use of custom ranking policies in the management search is not supported.

The search in the REST API for Content Delivery allows you to use a custom ranking policy as:

- **Global Default:** A ranking policy set as the global default overrides a built-in ranking policy and is applied by default to calculate the relevance score for items matching a search query.
- **Channel Default:** A ranking policy set as the channel default overrides a built-in ranking policy or the global default and is applied by default to calculate the relevance score for items matching a search query for this channel.
- **rankBy parameter value:** A ranking policy set as the value of the `rankBy` parameter on a search query request overrides any default policy and is applied to calculate the relevance score for items returned in the response.

Built-in Ranking Policy

The built-in ranking policy uses the following similarity ranking to calculate the relevance score for matched items:

- **Elasticsearch Relevance Score:** By default items matching a search query are ranked by the Elasticsearch similarity relevance score (the [Okapi BM25](#) algorithm).
- **Term Frequency in a Multi-Term Query:** For a multi-term search query, the Elasticsearch relevance score algorithm will take into account the term frequency (tf) when calculating the score for items matching either type-specific searches, or default (no type specified) searches.

- **Stemmed Matches:** The scoring algorithm will apply the same rank to non-stemmed and stemmed matches (for example, OCI compute topology and OCI compute topologies are equivalent). Within the group of matches with the same stem, the rank of non-stemmed matches is higher than the rank of stemmed matches.

Custom Ranking Policies

Custom ranking policies in Oracle Content Management allow you to change relevance scores of items matching a search query using the settings specified in the policy, independently from the structure of the submitted search query. That is achieved by giving you ability to define methods on a ranking policy to control:

- **Matching Rule:** What standard or user-defined fields on an asset type require matching (exact term - EQ or phrase match - MT) to given values for the ranking method to apply.
- **Score Change:** The impact a matching data field will have on the relevance score of an item (boost its original score or decrease it over time).

The two types of ranking methods available are boost and pin methods. These methods allow you to increase an item's relevance score, while decay methods decrease the score over a period of time.

To understand how ranking policy works, consider a simple example which uses items of the content type, automotive:

- Item 1: (name: Ferrari, description: Ferrari)
- Item 2: (name: Bugatti, description: Bugatti)

The following search queries will match and return both items, but the relevance scores calculated by the built-in ranking policy will be different. Thus, items will be returned in a different order with neither item potentially returned at the top by query 2.

- **Query 1:** `q=type eq "Automotive" and (description eq "Ferrari" or name eq "Bugatti")`
- **Query 2:** `q=type eq "Automotive"`

If you want search queries to always return items about Bugatti cars at the top (pinned) and items about Ferrari cars among the top matching items, you can define a custom ranking policy with two rules: boost - description (weight: 10, value: Ferrari) and pin - name (value: Bugatti). If this policy is used, either query will consistently return item 2 at the top and boost the relevance score of item 1 from the structure of the query:

- Query 1 result: Item 1 (Ferrari) boosted to 10 and Item 2 (Bugatti) is pinned
- Query 2 result: Item 1 (Ferrari) boosted to 10 and Item 2 (Bugatti) is pinned

Custom Ranking Policies Lifecycle

A new ranking policy is created as a draft version, which you can promote. Promoting a draft version of a ranking policy creates a promoted version. Users with the Content Administrator role can set a promoted ranking policy as the global default. A manager on a publishing channel can assign the promoted version as a channel default policy to it. Just as the localization policy, only one ranking policy can be assigned to a channel at a time.

To make a ranking policy available in the REST API for Content Delivery, it needs to be published to create a published version. Content administrators can publish a ranking policy on the ranking policies page.

The following table summarizes actions that are allowed on a ranking policy and the policy status in Content Preview or Delivery Search API depending on the available versions of the policy:

Available Version	Promote	Set as Channel Default	Set as Global Default	Publish	Unpublish	Delivery API
Draft	yes	no	no	no	n/a	no
Promoted	n/a	yes	yes	yes	n/a	no
Published	n/a	yes	yes	n/a	yes	active

 **Note:**

- Draft, promoted, and published versions have the same policy descriptor which uniquely identifies a ranking policy.
- The active status means that results matching a search query in the REST API for Content Delivery will be ranked by relevance based on the rules defined in the ranking policy if it is set as a global default, assigned as a channel default, or specified in the `rankBy` parameter on the request URL.

Supported Ranking Methods

A custom ranking policy in Oracle Content Management is a collection of ranking methods, each defined to match a standard or user-defined data field on asset types and then boost or decrease the relevance score of matching items. When a ranking policy is applied, an item matching a search query that you submitted can meet matching rules in one method, several methods, or none. If the item didn't match any method, its relevance score remains the same as calculated by the built-in ranking policy (Elasticsearch), original score. If an item matches one method, its original score is multiplied by the parameter weight value set on this method. If the item matches more than one ranking method, its relevance score = original score * weight1 * weight2 * ...

The following ranking methods are currently supported in Oracle Content Management for defining custom ranking policies:

Ranking Method	Method Parameters	Data Field Type Compatibility	Comment
Boost - Equals	asset type - field {1-n} value {1-m} weight	Text, Date, Numeric	If more than one value is set for a selected data field, OR is used in the expression to match items for boosting. The value of the date type is matched within a given time interval.

Ranking Method	Method Parameters	Data Field Type Compatibility	Comment
Boost - Phrase Match	asset type - field {1-n} value {1-m} weight	Text	If more than one value is set for a selected data field, OR is used in the expression to match items for boosting. A phrase value is matched anywhere in the field data. For example, if the value set on the method is Italy, it will match both "Italy is a European country" or "Rome is the capital of Italy" phrases.
Pin - Equals	asset type - field {1-n} value {1-m}	Text, Date, Numeric	Weight is a server-controlled value that is large enough to ensure pinned items bubble up on the top. If more than one value is set for a selected field, OR is used in the expression to match items for pinning.
Pin - Phrase Match	asset type - field {1-n} value {1-m}	Text	Weight is a server-controlled value that is large enough to ensure pinned items bubble up on the top. If more than one value is set for a selected field, OR is used in the expression to match items for pinning.
Decay - Date	asset type - field {1-n} origin offset scale	Date	Standard fields: Only publishedDate field or user-defined fields: Any field of the date type. Method decreases the existing relevance score of an item.
Decay - Numeric	asset type - field (1-n) origin offset scale	Numeric	User-defined fields: Any field of the numeric type. Method decreases existing relevance score of an item.

 **Note:**

- In **Typed searches**, that is, for search requests that explicitly specify asset types in the search query expression, Boost, Pin or Decay ranking methods will use **both standard and user-defined data fields** on asset types to match items for boosting, pinning, or decreasing their relevance score.
- In **Untyped or Default searches**, that is, for search requests that do not specify asset types in the search query expression, Boost, Pin or Decay ranking methods will use **only standard data fields** on asset types to match items for boosting, pinning, or decreasing their relevance score.
- **Multiple matching fields** : If a ranking method is configured to match several fields on an asset type, then the relevance score of the items that match several or all fields will be a multiple of weights set on the method. For example, if two fields in ranking method with weight 20 are matched, then the item score will be $20 \times 20 = 400$.

The Boost - Equals Method

The Boost - Equals method allows you to increase the relevance score of an item that matches the submitted search request so long as the item has the asset types set in the method and its fields match the exact value set in the method (the EQ operator is applied for matching the field value). The original relevance score of an item is multiplied by the weight defined in the method.

If the item matches more than one field defined on the Boost - Equals or Boost - Phrase Match method, its relevance score is calculated as follows:

- Item matches single field: Item score = original score * weight
- Item matches several fields: Item score = original score * weight * weight * ...

 **Note:**

The field value of the date type on an asset matches the value set in the Boost - Equals method within time interval from Value to Value + Offset.

Sample request

```
PUT /content/management/api/v1.1/search/rankingPolicies/
5A487437500042849B54FE3BA4EC80C2?q=(status%20eq%20%22draft%22)&fields=all
```

```
{
  "name": "rp1",
  "description": "",
  "apiName": "rp1",
  "entries": [
    {
      "key": {
        "name": "myBoostEquals",
        "methodType": "equal",
        "weight": 10,
```



```

    "values": [
      {
        "type": "text",
        "entries": [
          "cat"
        ]
      },
      {
        "type": "numeric",
        "entries": []
      },
      {
        "type": "datetime",
        "entries": [
          {
            "origin": "2022-05-10T00:00:00.000-07:00",
            "originTimeZone": "America/Los_Angeles",
            "offset": "10d"
          }
        ]
      }
    ],
    "type": "boostMethod"
  },
  "value": [
    {
      "name": "name",
      "contentType": "myType1",
      "weightMultiplier": 2,
      "type": "standardAssetField"
    },
    {
      "name": "published date",
      "contentType": "myType1",
      "weightMultiplier": 1,
      "type": "standardAssetField"
    }
  ]
},
"policyId": "B025517B9D47454A9311EDD3D4D92BD6"
}

```

The Boost - Phrase Match Method

The Boost - Phrase Match method allows you to increase the relevance score of an item that matches the submitted search request so long as the item has the asset types set in the method and its fields match the phrase value set in the method (the MT operator is applied for matching the field value). The original relevance score of the item is multiplied by the weight defined in the method.

Sample request

```

PUT /content/management/api/v1.1/search/rankingPolicies/
5A487437500042849B54FE3BA4EC80C?

```

```

q=(status%20eq%20%22draft%22)&fields=all

{
  "name": "rp1",
  "description": "",
  "apiName": "rp1",
  "entries": [
    {
      "key": {
        "name": "myBoostMethod1",
        "methodType": "phraseMatch",
        "weight": 10,
        "values": [
          {
            "type": "text",
            "entries": [
              "cat"
            ]
          }
        ]
      },
      "type": "boostMethod"
    },
    {
      "value": [
        {
          "name": "name",
          "contentType": "myType1",
          "weightMultiplier": 2,
          "type": "standardAssetField"
        },
        {
          "weightMultiplier": 10,
          "contentType": "myType1",
          "name": "myField1",
          "type": "customAssetField",
          "id": "2332EA112CD140F7A4D6847451B1355F"
        }
      ]
    }
  ],
  "policyId": "B025517B9D47454A9311EDD3D4D92BD6"
}

```

The Pin - Equals Method

The Pin - Equals method allows you to set the topmost relevance to an item that matches the submitted search request so long as the item has the asset types set in the method and its fields match the exact value set in the method (the EQ operator is applied for matching the field value).

If the item matches more than one field defined on the Pin - Equals or Pin - Phrase Match method, its relevance score is calculated as follows:

- Item matches single field: Item score = original score * 5000
- Item matches several fields: Item score = original score * 5000 * 5000 * ...

 **Note:**

The field value of the date type on an asset is matched to the value set in the Pin - Equals method within the time interval from Value to Value + Offset.

Sample request

```
PUT /content/management/api/v1.1/search/rankingPolicies/
5A487437500042849B54FE3BA4EC80C2?
q=(status%20eq%20%22draft%22)&fields=all
```

```
{
  "name": "rpl",
  "description": "",
  "apiName": "rpl",
  "entries": [
    {
      "key": {
        "values": [
          {
            "type": "text"
          },
          {
            "type": "numeric"
          },
          {
            "type": "datetime",
            "entries": [
              {
                "origin": "2022-05-10T00:00:00.000-07:00",
                "originTimeZone": "America/Los_Angeles",
                "offset": "12h"
              }
            ]
          }
        ]
      },
      "name": "myPinEquals1",
      "type": "pinMethod",
      "methodType": "equal"
    },
    "value": [
      {
        "contentType": "myType1",
        "name": "name",
        "type": "standardAssetField"
      },
      {
        "contentType": "myType1",
        "name": "published date",
        "type": "standardAssetField"
      }
    ]
  }
}
```

```

    }
  ],
  "policyId": "B025517B9D47454A9311EDD3D4D92BD6"
}

```

The Pin - Phrase Match Method

The Pin - Phrase Match method allows you to set the topmost relevance to the item that matches the submitted search request so long as the item has the asset types set in the method and its fields match the phrase value set in the method (the MT operator is applied for matching the field value).

Sample request

```

PUT /content/management/api/v1.1/search/rankingPolicies/
5A487437500042849B54FE3BA4EC80C2?q=(status%20eq%20%22draft%22)&fields=all

```

```

{
  "name": "rp1",
  "description": "",
  "apiName": "rp1",
  "entries": [
    {
      "key": {
        "values": [
          {
            "type": "text",
            "entries": [
              "cat",
              "black"
            ]
          }
        ]
      },
      "name": "myPinPhraseMatch1",
      "type": "pinMethod",
      "methodType": "phraseMatch"
    },
    "value": [
      {
        "contentType": "myType1",
        "name": "description",
        "type": "standardAssetField"
      },
      {
        "contentType": "myType1",
        "name": "name",
        "type": "standardAssetField"
      },
      {
        "contentType": "myType1",
        "name": "myfield1",
        "type": "customAssetField",
        "id": "8921A6FDFD1C4C2AB966DE0BA1FDF3D"
      }
    ]
  ]
}

```

```

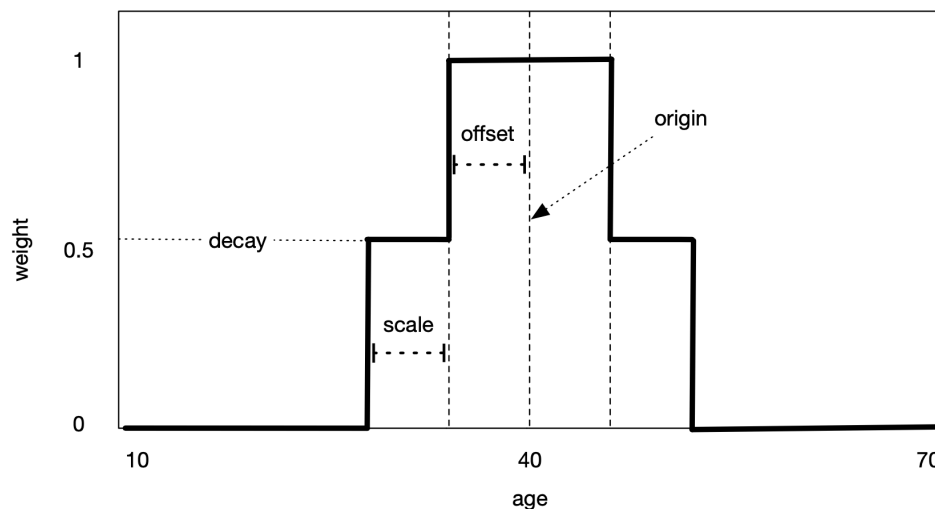
    }
  ],
  "policyId": "B025517B9D47454A9311EDD3D4D92BD6"
}

```

The Decay - Date Method

The Decay - Date method allows you to decrease the relevance of an item that matches the submitted search request so long as the item has the asset types set in the method. The relevance score that this item has at the origin date will decrease the further value of the date field selected on the asset type gets away from the origin.

Currently the Decay - Date or Decay - Numeric method in Oracle Content Management does not support Gaussian, exponential, or linear decay functions that are supported in Elasticsearch. Instead Oracle Content Management allows you to define a step-down decay function to achieve a similar effect:



The calculated relevance score of an item with the field value matching origin +/- offset is multiplied by the weight which is calculated by the Decay method. The relevance score of the matching item may be boosted by a Boost ranking method. If scale = 0, the weight drops to 0 whenever the field value is above origin + offset or below origin - offset values. Otherwise, the weight drops to half of the decay value for the duration of the scale value and then drops to 0.

Sample request

```

PUT /content/management/api/v1.1/search/rankingPolicies/
5A487437500042849B54FE3BA4EC80C2?
q=(status%20eq%20%22draft%22)&fields=all

```

```

{
  "name": "rp1",
  "description": "",
  "apiName": "rp1",
  "entries": [
    {
      "key": {
        "name": "myDecayDate",

```

```

        "type": "decayMethod",
        "origin": "2022-05-10T00:00:00.000-07:00",
        "originTimeZone": "America/Los_Angeles",
        "offset": "6h",
        "scale": "12h",
        "decay": "0.1",
        "methodType": "date"
    },
    "value": [
        {
            "contentType": "myType1",
            "name": "published date",
            "type": "standardAssetField"
        }
    ]
}
],
"policyId": "B025517B9D47454A9311EDD3D4D92BD6"
}

```

The Decay - Numeric Method

The Decay - Numeric method allows you to decrease the relevance of the item that matches the submitted search request so long as this item has the asset type set in the method. The relevance score that this item has initially will decrease the further value of the numeric field selected on the asset type gets away from the origin value.

The Decay - Numeric method uses the same step-down decay function as the Decay - Date method.

Sample request

```
PUT /content/management/api/v1.1/search/rankingPolicies/
5A487437500042849B54FE3BA4EC80C2?q=(status%20eq%20%22draft%22)&fields=all
```

```

{
  "name": "rp1",
  "description": "",
  "apiName": "rp1",
  "entries": [
    {
      "key": {
        "name": "myDecayNumeric",
        "type": "decayMethod",
        "origin": "10",
        "offset": "2",
        "scale": "5",
        "decay": "0.1",
        "methodType": "numeric"
      },
      "value": [
        {
          "contentType": "myType1",
          "name": "mynumberfield1",
          "type": "customAssetField",

```

```
        "id": "F0C4FA257D5B4C6C81656837D867B29E"  
      }  
    ]  
  }  
],  
  "policyId": "B025517B9D47454A9311EDD3D4D92BD6"  
}
```

Use REST APIs for Extended Workflow

The integration of Oracle Content Management (OCM) with Oracle Integration Cloud (OIC) – Process Automation Integration lets users manage business processes, such as content workflows to route content for approval or review. Oracle Content Management provides a quick start application package with several multistep processes that you can deploy to your OIC instance. You can use the processes for review and approvals of assets that you manage in the OCM asset repositories.

One of the processes, one-step content approval and publish with extended workflow, now includes an upgrade that allows users to track the asset publishing status after an asset is approved or rejected. When the asset is successfully published to its target channel, the workflow instance is complete when the process notifies OCM that the workflow is finished. Otherwise, the asset remains in the workflow, and users will be notified of any publishing errors and asked to take actions to complete the workflow.

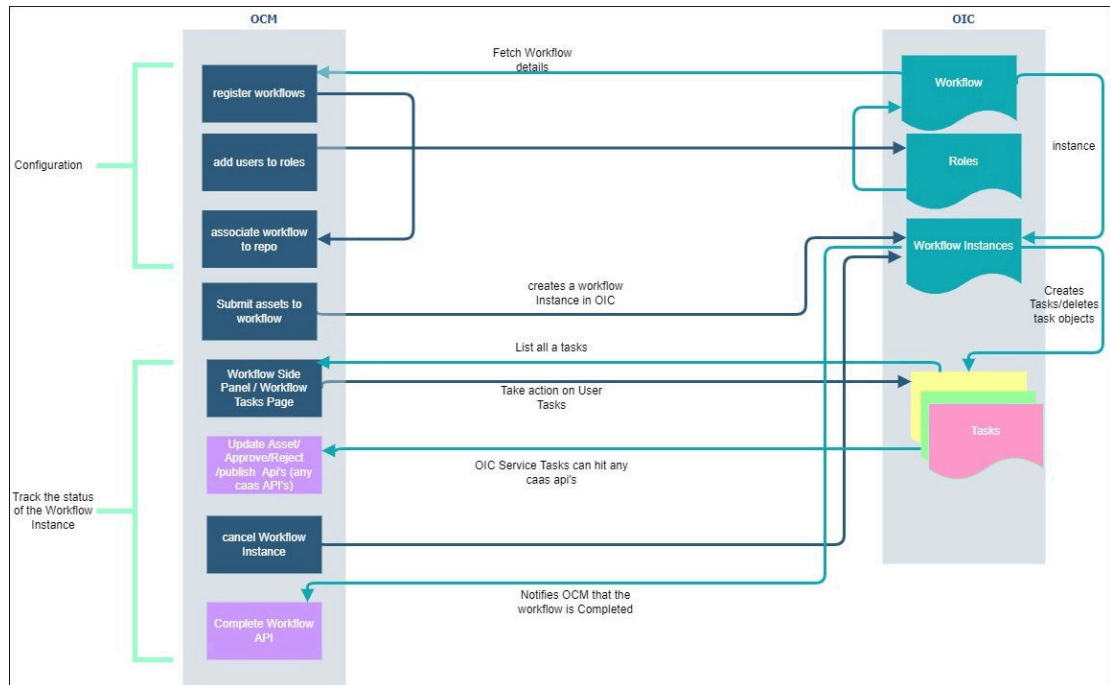


Note:

If an asset has dependencies, the parent asset must first complete the workflow before the child asset can be submitted to complete the workflow.

Complete Workflow Instance API

The image below shows that the new API `CompleteWorkflowInstance` provides users the ability to complete the workflow instance. To review the API details, see [Performs Bulk Items Operations](#) in the API reference guide.



Search In-Workflow Assets

Filtering assets that are In Workflow is enabled by the search API: `GET /content/management/api/v1.1/items`, which has introduced the new `workflowInstances` node in the search API response, stored as nested objects.

To filter the assets with the search API, use the workflow name or workflow id. For example, you can list all the assets that are In Workflow with the process name `OCEOneStepProcess`. Currently the search API returns only the latest workflow to which the asset is submitted.

<code>"workflowInstances":{</code>
<code> "data":[{</code>
<code> "Id":"E29EF1BDA8DC4F989DCDF89030069CDB",</code>
<code> "Name":"OCEOneStepProcess",</code>
<code> "isCompleted":false</code>
<code> },</code>
<code> {</code>
<code> "Id":"9EFB1AA41FCF404999AF0D6F07E55CDD",</code>
<code> "Name":"OCETwoStepProcess",</code>
<code> "isCompleted":true</code>
<code> }],</code>
<code> "links":[{</code>
<code> "href":"http://ssvrint-oracle.apmogili-1.cec.oraclecorp.com:8080/content/management/api/v1.1/items/CONT66F3A51EF11C4A51A9431A89E090016F/workflowInstances",</code>
<code> "rel":"self",</code>
<code> "method":"GET",</code>
<code> "mediaType":"application/json"</code>
<code> }]</code>
<code>]</code>
<code>}</code>
<code></code>

Create and Use Applinks for File and Folder Access

You can control access to files and folders in Oracle Content Management with applink endpoints of the REST API for Documents.

The following endpoints let you control user access to files and folders:

- Create File Applink
- Create Folder Applink
- Refresh Applink Token

To create an applink for a user to access a file or folder, you must be the owner or have the manager role. Either of these roles gives you admin privileges for the file or folder.

When you create a file or folder applink for a user, you can grant the user any of these roles:

- **Viewer:** Viewers can look at files and folders, but can't change things.
- **Downloader:** Downloaders can also download files and save them to their own computers.

- **Contributor:** Contributors can also modify files, update files, upload new files, and delete files.

The response from the Create File Applink or Create Folder Applink REST API includes an applink ID, access token, and refresh token. For example:

```
{
  "appLinkId": "LDhsn4VPTsnDnKpKLFZTCkjaPkYbMC6-3taYSdJAazckhezJ2H1Sjs2TH0ou6cCAvxcRnw5gpXcU7pIRkCmWN8kEToJHFwwZ-ptWvPGhJaiir19baL9mka14WnwpL6au0040-gFMPvkPv230tMnj2W3A==",
  "accessToken": "GYrSN5zuj0kOTE4k_60bKvdkxx2-ARA546A2T77GtEOgoPZPGgKk126OeCn1w-Ij",
  "appLinkUrl": "http://www.example.com/documents/embed/link/app/LDhsn4VPTsnDnKpKLFZTCkjaPkYbMC6-3taYSdJAazckhezJ2H1Sjs2TH0ou6cCAvxcRnw5gpXcU7pIRkCmWN8kEToJHFwwZ-ptWvPGhJaiir19baL9mka14WnwpL6au0040-gFMPvkPv230tMnj2W3A==/fileview/DFD11F62E911327CB1F160F6T0000000000100000001",
  "refreshToken": "Yc_b_dE8V03eDTCmcmC1gi_y3LVJTPiZOSQDhuS_VWim9E_FRpLQGtEhgxCNbKTG",
  "role": "manager",
  "id": "DFD11F62E911327CB1F160F6T0000000000100000001",
  "type": "applink",
  "errorCode": "0"
}
```

The user needs to pass the applink ID and access token parameters in the request header for each REST API operating on the folder or file or on any child folder or file. The response does not provide a link to the file or folder because headers cannot be represented in a link.

The access token expires after 15 minutes. You can refresh the access token any number of times within the time period defined by a refreshed token (24 hours). Use the refresh token to request a new access token when the current access token expires. The refreshed token expires after 24 hours.

For more information about the applink REST APIs, see "Applinks" in [REST API for Documents](#).

Provide Access to Files and Folders with Public Links

To give users access to files and folders, you can use public links to share the files and folders with assigned users.



Note:

Public links aren't supported in private instances.

You can create multiple, named links for the same file or folder resource with different access roles, expiration dates, and so on. You can also edit some of the information for an existing public link.

To provide access to a files and folders with a public links:

1. Use the Create File Public Link and Create Folder Public link REST APIs to create the public links you need.
2. If you want to edit the available public link parameters, use Edit Public Link REST API.
3. Use the Get File Public Link or Get Folder Public Link REST API to get information about the public links available for a file or folder.

For example, the follow Get Folder Public Link request specifies the folder ID F4E111D0D0645CD368453C2BT000000000100000001:

```
GET . . ./publiclinks/folder/
F4E111D0D0645CD368453C2BT000000000100000001
```

If the HTTP Status Code is 200 and one or more public links are available for the specified folder, the response returns information about public links defined for the folder:

```
{
  "count": "2",
  "errorCode": "0",
  "id": "F4E111D0D0645CD368453C2BT000000000100000001",
  "type": "folder",
  "items": [
    {
      "type": "publiclink",
      "linkID":
"LF31C09DE51854DBBDA37A90T000000000100000001",
      "linkName": "hasSecondLink",
      "ownedBy": {
        "displayName": "User AA",
        "loginName": "userAALoginName",
        "id": "U0EAA20910FAF3052ACB79E4T0000000001",
        "type": "user"
      },
      "role": "viewer",
      "assignedUsers": "@everybody",
      "createdTime": "2015-06-02T19:30:37Z",
      "lastModifiedTime": "2015-06-02T19:30:37Z"
    },
    {
      "type": "publiclink",
      "linkID":
"LF5E5F73A444FFB8924EF8ACT000000000100000001",
      "linkName": "hasFirstLink",
      "ownedBy": {
        "displayName": "User AA",
        "loginName": "userAALoginName",
        "id": "U0EAA20910FAF3052ACB79E4T0000000001",
        "type": "user"
      },
      "role": "contributor",
      "assignedUsers": "@serviceinstance",
      "createdTime": "2015-06-10T16:15:37Z",
      "lastModifiedTime": "2015-06-10T16:15:37Z"
    }
  ]
}
```

```

    }
  ]
}

```

This response shows that two public links are available, one for users who are assigned the viewer role for the folder and one for users who are assigned the contributor role. You use information from the response in API headers to share the folder with the assigned users (`assignedUsers` value) specified in the public link.

In the example, the second link grants contributor-level access to all account holders for the Content Management instance that hosts the specified folder.

For example:

```

@serviceinstance/documents/link/
LF5E5F73A444FFB8924EF8ACT0000000000100000001/folder/
F4E111D0D0645CD368453C2BT0000000000100000001

```

4. To share a file or folder, use the information from the response in the headers of other APIs, like Share Folder or Collection.

A web client can use a URL in one of the following formats to allow file or folder visualization through a public link:

```
domain URL/documents/link/linkID/fileview/file ID
```

```
domain URL/documents/link/linkID/folder/folder ID
```

For more information about public links, see Publiclinks Resource or "Publiclinks" in [REST API for Documents](#).

Upload a REST API Swagger File into Mobile Cloud Service

In Oracle Mobile Cloud Service (MCS), you can create a REST connector API to connect to the REST API for Documents.

If you provide a Swagger descriptor URL, the REST Connector API wizard can examine the descriptive metadata and obtain resources and fields from it.

Only Swagger metadata in JSON format is currently supported.

A REST connector API is a configuration for communicating with a specific external service to send and receive data. REST connector APIs give you a standard way to connect a mobile app to existing REST services and at the same time benefit from the Oracle Mobile Cloud Service's built-in security, diagnostics, and analytics features.

The connector API communicates and passes information between the client and the server using the HTTPS protocol. The information passed can be in the form of XML or JSON (JavaScript Object Notation). REST doesn't contain a messaging layer. It uses a set of rules to create a stateless service.

The REST Connector API wizard walks you through creating REST connector APIs, from specifying a remote service and setting security policies to testing your endpoints. Additionally, the wizard supports Swagger. When you provide a Swagger descriptor URL, the wizard introspects the endpoints available from that file. The available resources are identified and displayed. You simply select the resources of the external service.

Only standard internet access ports 80 and 443 are supported. Connection to a service can't be made using a custom port.

Use the REST Connector API wizard to quickly configure your connector API by providing a name and description, specifying timeout settings, adding rules, setting a security policy, and testing it.

Creating a connection to an existing REST service can be a simple two-step operation:

1. Name your REST connector API.
2. Provide the URL to the external service.

In the REST Connector API wizard, on the page where you would provide the URL to the Swagger file, there's a link to the internal Oracle API catalog. You can get the Swagger URL from there and paste it in the field provided for the descriptor.

You have the option of applying rules to form specific requests or responses for the data that you want to access. In addition, you have the ability to configure client-side security policies for the service that you're accessing and test and check the results of your connection.

As soon as it's created, your REST connector API is listed on the Connectors page. When at least one REST connector API exists, you'll be taken directly to the Connectors page when you click Connectors from the side menu. From the catalog, you can select the REST connector API and edit it, publish it, create a new version or update an existing version, deploy it if it has a Published state, or move it to the trash.

For more information, see "[REST Connector APIs](#)" in *Developing Applications with Oracle Mobile Cloud, Enterprise*.

Oracle Content Management APIs

The Oracle Content Management APIs allow you to work with functionalities in site building and embed user interface features of Oracle Content Management.

- [Embed UI API V2](#)
- [Site Compile API](#)
- [Sites Component API](#)
- [Sites Rendering API](#)

Embed UI API V2

The [Embed UI API V2 for Oracle Content Management](#) is a JavaScript API that enables you to embed the Oracle Content Management web user interface into your own web applications using an HTML inline frame (iframe tag). The JavaScript API simplifies the creation of the iframe element and manages communication with the code running in the frame. The embedded web interface can include asset and document lists, conversations, site content, search results, and other Oracle Content Management features.

Site Compile API

The [Site Compile API](#) for Oracle Content Management allows you to compile sites, which creates static, server-side rendered HTML files for all pages in a site.

Sites Component API

The [Sites Component API](#) (SCSComponentAPI) is an environment agnostic object present on all Oracle Content Management web pages and during Site Compilation. It is primarily responsible for providing information about the site to allow developers to create HTML for components that can be inserted into the page.

Sites Rendering API

The [Sites Rendering API](#) (SCSRenderAPI) is a window-global object present on all Oracle Content Management web pages. It is primarily responsible for rendering the slots and components of the page, and it provides an interface for JavaScript code present on theme layouts. The runtime SCSRenderAPI renders the view and preview display modes. The design time SCSRenderAPI renders navigate, edit, and annotation display modes.

Oracle Content Management SDKs

Oracle Content Management provides software development kits (SDKs) that help you integrate Oracle Content Management functionality and simplify your application development:

- [Content SDK for JavaScript](#)
- [Content SDK for Java](#)
- [Content SDK for Swift](#)
- [Sites SDK](#)
- [Translation Connector SDK](#)

Content SDK for JavaScript

The Content SDK for Oracle Content Management is a light-weight JavaScript wrapper that interacts with the Content REST APIs.

This read-only SDK retrieves structured content, digital assets, and content layouts that are managed in Oracle Content Management. The SDK can be used in web browsers or NodeJS projects.

The Content SDK for JavaScript consists of three main modules:

- **ContentSDK**: The main entry-point object. The ContentSDK object lets you create client objects to access content based on your requirements.
- **ContentDeliveryClient** : A client object that is set up to access published content items and digital assets.
- **ContentPreviewClient** : A client object that is set up to access content types, draft content items, and draft digital assets.

The Content SDK for JavaScript is available as an Oracle open-source project on [GitHub](#).

The reference guide can be found here, [Content SDK for JavaScript](#).

Content SDK for Java

Oracle Content Management provides a Content SDK for Java/Android. The read-only SDK is a package of libraries for retrieving published content items, digital assets, and content layouts that are managed in Oracle Content Management.

The SDK is a light-weight Android binding that interacts with the [REST API for Content Delivery](#). The SDK can easily be integrated with any third-party Android mobile application. The SDK lets you fetch content from the server on the fly, without the need for rebuilding the app to modify content. The SDK also provides a wide range of advanced utilities and features such as response caching, a search request builder, and request/response modeling.

The Content SDK for Java is available as an Oracle open-source project on [GitHub](#).

The reference guide can be found here, [Content SDK for Java/Android](#).

 **Note:**

The SDK will work in stand-alone Java applications as it does not contain any direct dependencies on the Android SDK; however, most examples are shown as they would be coded in an Android application as it is assumed that is the primary target platform. This SDK will also work for Android apps written in Kotlin.

Content SDK for Swift

Oracle Content Management provides a Content SDK for [Swift](#), which is a powerful and intuitive programming language for iOS, iPadOS, macOS, tvOS, and watchOS. The read-only SDK is a package of libraries for retrieving published content items, digital assets, and content layouts that are managed in Oracle Content Management.

The SDK is a light-weight iOS binding that interacts with the [REST API for Content Delivery](#), and can be easily integrated with any third-party iOS mobile application. The SDK lets you fetch content from the server on the fly, without the need for rebuilding the app to modify content. The SDK also provides a wide range of advanced utilities and features such as response caching, a search request builder, and request/response modeling. There are two main libraries:

- **OracleContentCore**, which provides base functionality, including network transport capabilities, that's shared and required by multiple Oracle Content libraries.
- **OracleContentDelivery**, which provides data models and APIs for consuming content from Oracle Content Management.

The Content SDK for Swift is available as an Oracle open-source project on [GitHub](#).

The reference guides can be found here:

- [Content SDK for Swift/iOS \(Core\)](#)
- [Content SDK for Swift/iOS \(Delivery\)](#)

Sites SDK

The Sites SDK for Oracle Content Management is a JavaScript library that provides a set of functions which enable components to have a more integrated experience with Oracle Content Management.

The [Sites SDK](#) is available for download from the Oracle Content Management server:

`http://{server}/_sitesclouddelivery/renderer/app/sdk/js/sites.min.js`

Translation Connector SDK

The Translation Connector SDK for Oracle Content Management is a sample NodeJS implementation of the [Translation Connector REST API](#). The sample accepts an

Oracle Content Management translation job zip file, translates all the resources in the file, and returns a new zip file containing all the translations.

The Translation Connector SDK is part of Content Toolkit, which is available on [GitHub](#).

The reference guide can be found here, [Translation Connector SDK](#).

GraphQL Support in Oracle Content Management

Oracle Content Management includes support for GraphQL for all published content and all asset types. Developers can inspect the schema, build queries, and invoke them from clients alongside other delivery APIs.

GraphQL is a well-known query language for accessing content that is focused on fetching only the content that clients request and nothing more. GraphQL is designed to make APIs flexible and developer-friendly. To support this flexibility and power, GraphQL includes complete and inspectable schema that developers can build queries against in an intuitive way. For more information, visit <https://graphql.org/> and <https://spec.graphql.org/October2021/>

The following sections describe how to work with GraphQL in Oracle Content Management:

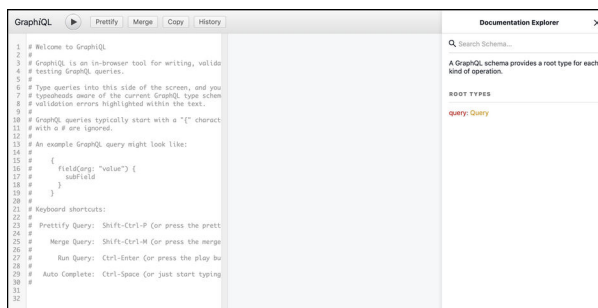
- [Get Started with GraphQL](#)
- [GraphQL Schema](#)
- [GraphQL Queries](#)
- [GraphQL Support for Content Preview](#)
- [GraphQL Samples](#)

Get Started with GraphQL

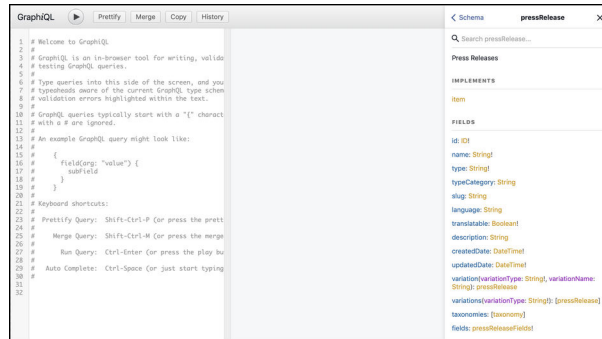
The quickest way to get started is by exploring schema and trying a few queries. Oracle Content Management includes a GraphQL IDE that helps you explore schema and run queries.

Explore Schema

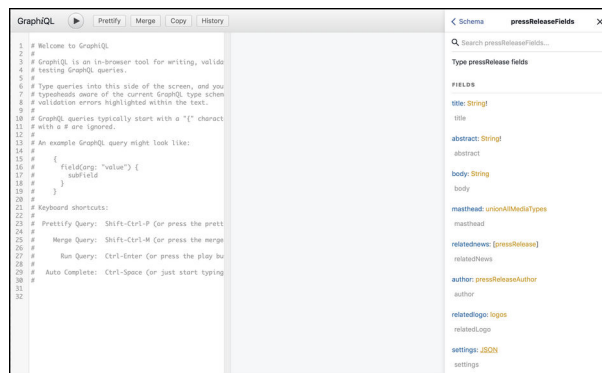
The GraphQL IDE is available at http://your_instance/content/published/api/v1.1/graphql/explorer.



In the Documentation Explorer (top right), search for an asset type by name to view the schema for that type. The image below shows an asset type named *PressRelease*. Every asset type in Oracle Content Management is represented by a GraphQL type of the same name (applying some helpful naming convention conversions). Every field in the asset type is represented by a GraphQL field with an equivalent data type.




View the details of related types by clicking on the type's name. For example, all user defined fields of *PressRelease* are in the *pressReleaseFields* type (the last entry in the schema shown above). The image below shows the details you see after clicking *pressReleaseFields*.



Try Some Queries

GraphQL provides access to content through queries. Use the queries in this section, in [GraphQL Queries](#), and in [GraphQL Samples](#) as examples.

To perform a content query, you need the IDs (or slugs) of a few published content items and the channelToken of the publishing channel. You type your query in the left panel, then click .

In the following example, we run a query with these parameters:

- Asset type: PressRelease
- Content item ID: CORE1FADA80EEACE4B4A84B76C07A931B317
- Publishing channel token: 573ae0bcb95347d283cdbea8a4d29641

```
{
  getPressRelease(id: "CORE1FADA80EEACE4B4A84B76C07A931B317",
    channelToken: "573ae0bcb95347d283cdbea8a4d29641") {
    id
    name
    description
  }
}
```

After running the query, you see the following.

```

GraphQL
1: {
2:   getPressRelease(id: "CORE1FADMBEACE4848487C07A931B327",
3:     channelId: "575a0bc995347d283c0e04d20641") {
4:     id
5:     name
6:     description
7:     fields {
8:     }
9:   }
10: }

```

```

1: {
2:   "data": {
3:     "getPressRelease": {
4:       "id": "CORE1FADMBEACE4848487C07A931B327",
5:       "name": "Red Bath & Beyond Selects Oracle to Modernize Enterprise Resource Planning",
6:       "description": "Red Bath & Beyond Selects Oracle to Modernize Enterprise Resource Planning and Accelerate Technology Transformation",
7:       "fields": {
8:       }
9:     }
10:   }
11: }

```

You can include additional fields in your query, such as user defined fields, references to other assets, and so on. The IDE helps you along in your query by automatically filling in field names.

```

GraphQL
1: {
2:   getPressRelease(id: "CORE1FADMBEACE4848487C07A931B327",
3:     channelId: "575a0bc995347d283c0e04d20641") {
4:     id
5:     name
6:     description
7:     fields {
8:       field {
9:         title
10:        abstract
11:        body
12:      }
13:    }
14:  }
15: }

```

```

1: {
2:   "data": {
3:     "getPressRelease": {
4:       "id": "CORE1FADMBEACE4848487C07A931B327",
5:       "name": "Red Bath & Beyond Selects Oracle to Modernize Enterprise Resource Planning",
6:       "description": "Red Bath & Beyond Selects Oracle to Modernize Enterprise Resource Planning and Accelerate Technology Transformation",
7:       "fields": [
8:         {
9:           "title": "Red Bath & Beyond Selects Oracle to Modernize Enterprise Resource Planning and Accelerate Technology Transformation",
10:          "abstract": "Red Bath & Beyond Inc. (NYSE: BBY) today announced it has selected Oracle as its Enterprise Resource Planning (ERP) technology provider. Oracle Cloud will provide real-time financial, supply chain and merchandising solutions, replacing the Company's legacy suite of technology systems and delivering new data, insights and planning capabilities.",
11:          "body": "Red Bath & Beyond Inc. (NYSE: BBY) today announced it has selected Oracle as its Enterprise Resource Planning (ERP) technology provider. Oracle Cloud will provide real-time financial, supply chain and merchandising solutions, replacing the Company's legacy suite of technology systems and delivering new data, insights and planning capabilities. Additionally, the agile partnership will enable continual innovation and improvement on our enterprise-wide, multi-ERP deployment is the first key component in the Company's $250 million technology investment roadmap to deploy industry-leading solutions that enhance the experience for customers and drive efficiencies across the enterprise. These technology investments will enable the use of analytics and automation to support improvements in merchandising and inventory management, product life cycle management, retail space planning and optimization, the launch of an array of exciting brand brands, and real-time tracking of merchandise fulfillment within the supply chain. Oracle Cloud ERP will also provide real-time financial and operational insights to support strategic planning decisions. \"Moving the right retail technology in place is fundamental to Red Bath & Beyond's transformation strategy and Oracle is proud to be supporting them in this journey,\" said Mike Webster, Senior Vice President and General Manager, Oracle Retail. \"By adopting Oracle Cloud, Red Bath & Beyond will be better able to manage its continually evolving inventory, plan margins, and improve sales forecasting in a digital-first shopping environment.\""}
12:        ]
13:       }
14:     }
15:   }
16: }

```

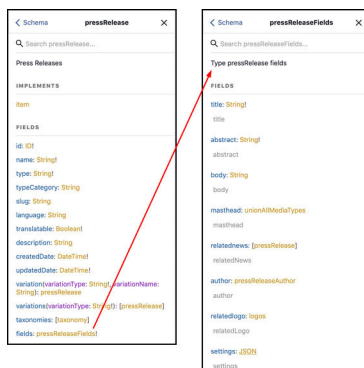
GraphQL Schema

GraphQL queries work against a schema or data model of the content. With Oracle Content Management, you don't need to create GraphQL schemas; every asset type in Oracle Content Management is automatically represented in the schema. The GraphQL type names are based on the API names of the asset types in Oracle Content Management, with prefixes where needed to avoid conflicts and collisions with reserved words in GraphQL.

Let's examine the schema for our *PressRelease* asset type in detail.

Types, Unions, and Interfaces

As we saw previously, the *PressRelease* asset type is represented by two GraphQL types—*pressRelease* and *pressReleaseFields*. The *pressRelease* type represents the core structure of the *PressRelease* asset type, including standard fields such as *id*, *name*, and *description*. The *pressReleaseFields* type, referenced in the *pressRelease* *fields* field, includes all user-defined fields. This structure mirrors how REST APIs express the field data.

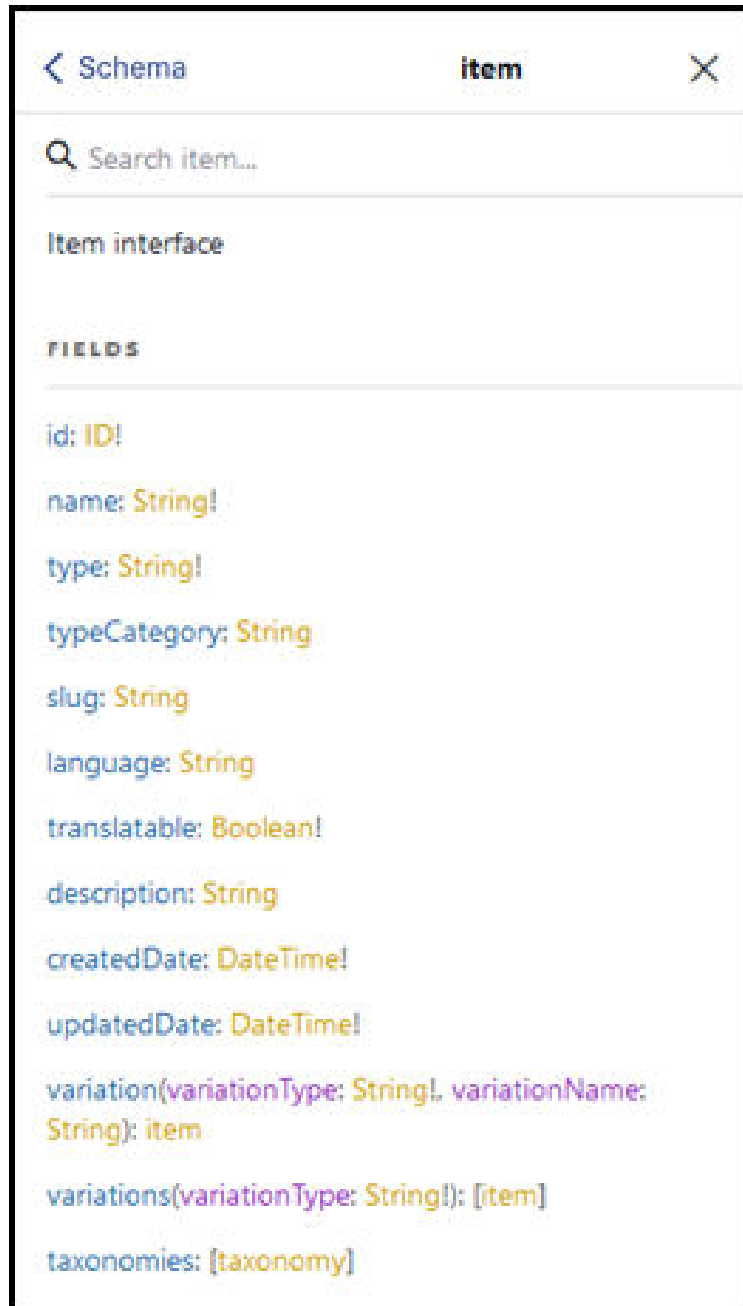


In addition to the user-defined fields (such as *title* and *abstract*), the *pressReleaseFields* type includes the associated data types (see the mapping below). If the fields are set as required in the asset type, the GraphQL schema shows these fields as required (for example *title*):

String!). If the fields are reference fields (Reference or Media), the GraphQL schema references the target's core asset type. For example, the *author* field references the *pressReleaseAuthor* asset type.

Reference fields in Oracle Content Management can accept more than one type. In such a scenario, a GraphQL union type is defined to represent such references. For example, the *masthead* field is declared with *unionAllMediaTypes*.

Notice that the *pressRelease* type implements an item; this is an interface item. The interface item (shown below) is the base for all core asset types in GraphQL. It includes access to taxonomies (taxonomies field) and translations (variation and variations fields). We'll discuss this more later.



The table below shows the item interface fields.

Schema item	
id:	ID!
name:	String!
type:	String!
typeCategory:	String
slug:	String
language:	String
translatable:	Boolean!
description:	String
createdDate:	DateTime!
updatedDate:	DateTime!
variation(variationType: String!, variationName: String!):	item
variations(variationType: String!):	[item]
taxonomies:	[taxonomy]

Data Types


Oracle Content Management data types are mapped to equivalent GraphQL types, native types where appropriate and scalars when needed. DateTime and JSON are scalars included automatically in the schema.

Oracle Content Management Data Type	GraphQL Field Type
Boolean	Boolean
Date	DateTime
Decimal	Float
JSON	JSON
Large Text	String
Media	target asset type or union
Number	Int
Reference	target asset type or union
Text	String

In addition to these data types, the Oracle Content Management ID for each asset is mapped to the ID field type in GraphQL.

GraphQL Queries

Oracle Content Management automatically generates queries for accessing assets. There is a generic item query (*getItem(..)*), and each core asset type gets a corresponding query (for example, *getPressRelease(..)*).

As previously mentioned, you type your query in the left panel, then click  The IDE will help you write your query, but here is the basic format you'll use to get an item, replacing the text in italics with the appropriate values. You *must* include a channelToken, either as an

argument in the query or as a header in the HTTP request. You must include either the ID or slug of the asset.

```
{
  getItem(channelToken: channel_token, id: ID, slug: slug): item
}
```

or

```
{
  getPressRelease(channelToken: channel_token, id: ID, slug: slug):
  pressRelease
}
```

In addition to queries that fetch a single asset, GraphQL also supports fetching many assets together, optionally based on a filtering criteria and sort order. In order to do that an out of the box query is included in the schema.

```
{
  getItems(channelToken: String,
  filter: standardFilter,
  query: queryFilter,
  sort: [standardSort],
  limit: int, offset: int) : itemCollection
}
```

A few standard types are introduced to support this functionality:

- `standardFilter` offers support for building filtering criteria for assets.
- `queryFilter` supports searching across many fields.
- `standardSort` specifies the sort order results based on field values.
- `limit` and `offset` provides means for client to paginate results.

The results of this query is itself a type that is a collection of other items, `itemCollection`, which in turn has an items array, along with pagination details.

The `standardFilter` type is defined in the schema as an input type.

< item standardFilter
🔍 Search standardFilter...
Type standard filter
FIELDS
id: idFilter
name: stringFilter
description: stringFilter
type: stringFilter
typeCategory: stringFilter
slug: stringFilter
language: stringFilter
translatable: booleanFilter
createdDate: dateTimeFilter
updatedAt: dateTimeFilter
AND: [standardFilter]
OR: [standardFilter]
NOT: standardFilter

The table below shows the standardFilter fields.

item standardFilter	
id:	idFilter
name:	stringFilter
description:	stringFilter
type:	stringFilter
typeCategory:	stringFilter
categories:	assetCategoryFilter
slug:	stringFilter
language:	stringFilter
translatable:	booleanFilter
createdDate:	dateTimeFilter
updatedAt:	dateTimeFilter
AND:	[standardFilter]
OR:	[standardFilter]

item standardFilter
NOT: standardFilter

Field names in this type are the same the names of the standard fields in Oracle Content Management for the item. Values accepted for these fields are themselves an input type, such as `stringFilter`, `idFilter` and others. These input types are specific to the underlying data type of the field, which represents a tuple of operation and value. The `stringFilter` type is expanded below to show the shape of this tuple.

<p>← <code>pressReleaseFilter</code> stringFilter ✕</p> <hr/> <p>🔍 Search stringFilter...</p> <hr/> <p>Type string filter</p> <hr/> <p>FIELDS</p> <hr/> <p>op: <code>stringFilterOps!</code></p> <p>value: <code>String!</code></p>	<p>← <code>stringFilter</code> stringFilterOps ✕</p> <hr/> <p>String filter options</p> <hr/> <p>VALUES</p> <hr/> <p><code>EQUALS</code></p> <p><code>NOT_EQUALS</code></p> <p><code>CONTAINS</code></p> <p><code>NOT_CONTAINS</code></p> <p><code>STARTS_WITH</code></p> <p><code>MATCH</code></p> <p><code>SIMILAR</code></p>
---	---

With these data-type-specific input fields, a query can include only valid criteria for the data type. Please see [GraphQL data types and available operations table](#) for details.

In addition to the generic item query (`getItems(. .)`) which uses the `standardFilter` type, GraphQL also supports type-specific queries with the ability to specify type-specific fields. To do this, for every asset type defined in Oracle Content Management, an accompanying type-specific filter is included. Continuing with the example of a `PressRelease` asset type, the filter for that is expressed as follows:

< Schema pressReleaseFilter ✕	< pressReleaseFilter pressReleaseFieldsFi... ✕
🔍 Search pressReleaseFilter...	🔍 Search pressReleaseFieldsFilter...
Type pressRelease filter	Type pressRelease fields filter.
FIELDS	FIELDS
id: <code>idFilter</code>	title: <code>stringFilter</code>
name: <code>stringFilter</code>	abstract: <code>largeTextFilter</code>
description: <code>stringFilter</code>	body: <code>largeTextFilter</code>
type: <code>stringFilter</code>	relatednews: <code>pressReleaseFilter</code>
typeCategory: <code>stringFilter</code>	author: <code>pressReleaseAutherFilter</code>
slug: <code>stringFilter</code>	relatedlogo: <code>logosFilter</code>
language: <code>stringFilter</code>	
translatable: <code>booleanFilter</code>	
createdAt: <code>dateTimeFilter</code>	
updatedAt: <code>dateTimeFilter</code>	
AND: [<code>pressReleaseFilter</code>]	
OR: [<code>pressReleaseFilter</code>]	
NOT: <code>pressReleaseFilter</code>	
fields: <code>pressReleaseFieldsFilter</code>	

Note that all fields present in the `standardFilter` type are available in the `pressReleaseFilter` type as well. This means, any query that is possible with the `standardFilter` type can also be built with the type-specific filter, but you can specify user-defined, field-specific filters as well. As with the `standardFilter` type, type-specific filters can be combined using `AND`, `OR`, and `NOT`.

Also note that type-specific fields include a field called `fields` that references another input type listing all user defined fields.

In addition to retrieving item content, the GraphQL schema also supports retrieving published taxonomy information using the following queries.

```
{
  getTaxonomy(channelToken: String, id: ID): taxonomy
}
```

and

```
{
  getTaxonomies(channelToken: String, sort: [taxonomySort]) :
```

```
taxonomyCollection
}
```

Categories can be fetched either from a taxonomy or directly, using the following queries.

```
{
  getCategory(id: ID, taxonomyId: ID, channelToken: String, apiName:
String): category
}
```

```
{
  getCategories(id: ID, taxonomyId: ID, channelToken: String, limit:
int, offset: int): categoryCollection
}
```

Examples

Retrieve Assets: Filter and Sort Assets by Standard Fields

Retrieving assets by filtering using standard fields (name, id, description, type, and so on) is accomplished by the use of `standardFilter`. For example, retrieving assets of a specific type is accomplished by `standardFilter` using criteria based on the type field's value.

```
{ getItems(channelToken: "573ae0bcb95347d283cdbea8a4d29641", filter: {type: {op:
EQUALS, value: "MyType"}}) { totalResults items { id name description } }
```

You could just as easily retrieve assets of more than one type by creating an OR condition in the `standardFilter`.

```
{
  getItems(channelToken: "573ae0bcb95347d283cdbea8a4d29641",
    filter: {
      OR: [
        {type: {op: EQUALS, value: "myType1"}},
        {type: {op: EQUALS, value: "myType2"}}
      ]
    }
  )
  {
    totalResults
    items {
      id
      name
      description
    }
  }
}
```

Note that filtering is possible on any standard fields present in the `Item` interface. Each field offers various operations (such as `EQUALS` and `NOT_EQUALS`) depending on the data type. See the following table for available operations for each data type.

In addition to filtering, clients can specify sorting criteria for the query. The following example sorts the results by the `name` field's value.

```
{
  getItems(channelToken: "573ae0bcb95347d283cdbea8a4d29641",
    filter: {type: {op: EQUALS, value: "MyType"}}
    sort: {name :DESC}
  ) {
    totalResults
    items {
      id
      name
      description
    }
  }
}
```

In addition to the filtering and sorting, clients can specify pagination instructions by using `limit` and `offset`. The following example shows how a client can fetch up to 10 results (`limit`) after the first 20 (`offset`).

```
{
  getItems(channelToken: "573ae0bcb95347d283cdbea8a4d29641",
    filter: {type: {op: EQUALS, value: "myType"}}
    sort: {name :DESC}
    limit: 10
    offset: 20
  ) {
    totalResults
    count
    items {
      id
      name
      description
    }
  }
}
```

Retrieving Assets of a Specific Type by Specifying a Type-Specific Filter

The filtering criteria in the following example includes a user-defined field `title` and queries for all `PressRelease` assets that have *Oracle* in the `title` field.

```
{
  getPressReleaseCollection(channelToken:
"573ae0bcb95347d283cdbea8a4d29641",
    filter: {fields: {title: {op: CONTAINS, value: "Oracle"}}}) {
    items {
      name
      id
    }
  }
}
```

As with any filters, type-specific filters can be nested.

```
{
  getPressReleaseCollection(channelToken:
"573ae0bcb95347d283cdbea8a4d29641",
  filter: {fields: {title: {op: CONTAINS, value: "Oracle"}}, AND:
{language: {op: EQUALS, value: "en-US"}}}) {
  items {
    name
    id
    language
    fields {
      title
    }
  }
}
```

Retrieve Taxonomy and Category Information

Just as in a single asset retrieval, taxonomy can be retrieved using `getTaxonomy`. The following example shows how a published taxonomy, along with the first 3 levels of category information, can be retrieved.

```
{
  getTaxonomy(channelToken: "573ae0bcb95347d283cdbea8a4d29641", id:
"F95CD0EB0E02427C9A3B7F1B8F33645C") {
    id
    name
    categories {
      id
      name
      children {
        id
        name
        children {
          id
          name
        }
      }
    }
  }
}
```

Likewise, it is possible to retrieve every published taxonomy by using `getTaxonomies`. The following example shows how to get every published taxonomy with first 3 levels of categories under each of them.

```
{
  getTaxonomies(channelToken: "573ae0bcb95347d283cdbea8a4d29641") {
    taxonomies {
      id
      name
      categories {
```

```

    id
    name
    children {
      id
      name
      children {
        id
        name
      }
    }
  }
}

```

Categories can themselves be listed as well as fetched by their `id/apiName`. The following example shows how to fetch a category by `apiName` and its children two levels down.

```

{
  getCategory(channelToken:"573ae0bcb95347d283cdbea8a4d29641", taxonomyId:
  "F95CD0EB0E02427C9A3B7F1B8F33645C" apiName:"hk-p")
  {
    name
    id
    apiName
    children {
      name
      id
      apiName
    }
  }
}

```

GraphQL Data Types and Available Operations	Values
String	EQUALS NOT_EQUALS CONTAINS NOT_CONTAINS STARTS_WITH MATCH SIMILAR
ID	IS
DateTime	EQUALS NOT_EQUALS

GraphQL Data Types and Available Operations	Values
	BEFORE AFTER BEFORE_OR_EQUAL AFTER_OR_EQUAL
Boolean	(no operation, just true/false)
Int and Float	EQUALS, NOT_EQUALS, GREATER_THAN

GraphQL Support for Content Preview

GraphQL supports querying unpublished content similar to Content Preview REST APIs. See [REST API for Content Preview](#).

Content that is published to a channel or targeted to a channel is available through Content Preview API in GraphQL. GraphQL preview provides access to the latest version of the asset.

In order to access assets in the preview context, the GraphQL endpoint is at `http://user_instance/content/preview/api/v1.1/graphql` and the GraphQL IDE is available at `http://user_instance/content/preview/api/v1.1/graphql/explorer`.

Previewing assets in GraphQL requires user authentication. See [Authorization](#) in *REST API for Content Preview*.

All the GraphQL capabilities (queries and schema) available for published assets are also available in preview.

GraphQL Samples

Retrieving An Asset's Data Using ID or Slug

The following query retrieves an asset's data using the asset's ID.

```
{
  getPressRelease(channelToken: "573ae0bcb95347d283cdbea8a4d29641",
id: "CORE1FADA80EEACE4B4A84B76C07A931B317")
  {
    id
    fields {
      title
      abstract
      body
    }
  }
}
```

```
    }  
  }  
}
```

The following query produces identical results to the one above, using the slug instead of the ID.

```
{  
  getPressRelease(channelToken: "573ae0bcb95347d283cdbea8a4d29641", slug:  
    "bed-bath-beyond-selects-oracle-to-modernize-enterprise")  
  {  
    id  
    fields {  
      title  
      abstract  
      body  
    }  
  }  
}
```

Retrieving a Referenced Asset's Data

The following query retrieves the referenced author type's fields along with the pressRelease fields.

```
{  
  getPressRelease(channelToken: "573ae0bcb95347d283cdbea8a4d29641", slug:  
    "bed-bath-beyond-selects-oracle-to-modernize-enterprise") {  
    id  
    fields {  
      title  
      abstract  
      body  
      author {  
        fields {  
          firstname  
          lastname  
          bio  
        }  
      }  
    }  
  }  
}
```

Retrieving Renditions of Digital Assets

An out-of-the-box type called *rendition* is included in the schema to represent various media renditions. It provides easy access to URLs of the renditions so you can easily include them in your code. For example, the following query returns a large rendition of an image.

```
{  
  getPressRelease(channelToken: "573ae0bcb95347d283cdbea8a4d29641", slug:  
    "bed-bath-beyond-selects-oracle-to-modernize-enterprise") {  
    id  
    fields {
```



```
        title
        abstract
        body
        masthead {
            ...headerImage
        }
    }
}
}
}

fragment headerImage on image {
  id
  fields {
    rendition(name: "Large", format: "jpg") {
      file {
        url
      }
    }
  }
}
}
```

Instead of retrieving a specific rendition, you can retrieve a list of renditions (with both jpg and webp formats) with the *renditions* parameter.

```
{
  getPressRelease(channelToken: "573ae0bcb95347d283cdbea8a4d29641",
slug: "bed-bath-beyond-selects-oracle-to-modernize-enterprise") {
    id
    fields {
      title
      abstract
      body
      masthead {
        ...headerImages
      }
    }
  }
}

fragment headerImages on image {
  id
  fields {
    renditions {
      file {
        url
      }
    }
  }
}
}
```

18

Use Webhooks

Oracle Content Management provides incoming webhooks you can use to receive event notifications from external applications.

You can also use outgoing webhooks to automatically push notifications of asset publishing, asset lifecycle, site publishing, prerendering, and scheduled jobs events to other applications.

The following topics describe how to use webhooks:

- [Outgoing Webhooks](#)
- [Incoming Webhooks](#)

Outgoing Webhooks

You can use outgoing webhook endpoints for push notifications of asset publishing, asset lifecycle, site publishing, prerendering, and scheduled jobs events.

Configure Outgoing Webhooks

You can use outgoing webhooks to have Oracle Content Management automatically post notifications based on asset and site events.

For example, you might want to post a notification to an application when new content is published, so that the application can consume the new content.

To create an outgoing webhook:

1. After you sign in to the Oracle Content Management web application as an administrator, click **Integrations** in the **Administration** area of the navigation menu on the left.
2. In the **Integrations** menu, click **Webhooks**.
3. Click **Create**.
4. Select the type of webhook you want to create:
 - **Asset Lifecycle Webhook:** Receive push notifications about asset lifecycle changes in a repository.
 - **Asset Publishing Webhook:** Receive push notifications about assets published to a channel or unpublished from a channel.
 - **Site Publishing Webhook:** Receive push notifications about site publishing.
 - **Prerender Page Refresh Webhook:** Receive push notifications about the need to render and cache a site detail page.
 - **Scheduled Jobs Webhook:** Receive push notifications about publishing jobs scheduled in a repository.
5. Enter a name and description for the webhook. This name appears as a heading for the posted conversation message.
6. Enable the webhook.

7. Provide information for webhook monitoring:
 - If you're creating an asset lifecycle webhook, select the repository you want to monitor for asset events. You can filter the notifications by [Asset Types](#), which applies only to events published for content items or digital assets of chosen types.
 - If you're creating an asset publishing webhook, select the publishing channels you want to monitor for asset publishing events.
 - If you're creating a site publishing webhook, specify the name of the site you want to monitor for site events.
 - If you're creating a prerender webhook, notify Prerender about the need to render and cache a Detail page.
 - If you're creating a scheduled jobs webhook, select the repository you want to monitor for scheduled jobs events.
8. Select the events you want to generate notifications.
9. Select whether you want brief or detailed information about each event.
 - For asset lifecycle webhooks, brief notifications include who triggered event, what was the event, what item was involved with the event, in which repository the event occurred, and when the event occurred. Detailed notifications include the same information listed in brief plus a list of all the fields in each content item or digital asset.
 - For asset publishing webhooks, brief notifications include the ID of each published asset. Detailed notifications include the same information listed in brief notifications plus a list of all the fields in each published asset.
10. Enter the target URL (endpoint) to which you want to post notifications.
11. If the endpoint requires authentication, select the type of authentication, and click **Details** to enter the authentication information.

Oracle Content Management (OCM) webhooks support the following options to configure authentication for the webhook notification receiver:

 - **None:** The receiver does not require authentication.
 - **Basic:** The receiver requires Basic Auth.
 - **Header:** The receiver requires a Secure Header.
12. To allow endpoints to use Self-Signed SSL certificates, which is recommended only in testing or development, select the corresponding checkbox:
 - **Self-signed SSL certificate:** The receiver accepts a self-signed SSL certificate.
13. To authenticate the call between your service and Oracle Content Management when event notifications are sent to your webhook endpoints, OCM signs the event with a signature. The signature is a security token in hash-based message authentication (HMAC) code, using the standard SHA-256 HMAC cryptography. You can use the HMAC token to verify that the notifications are sent by Oracle Content Management.

Enable the signature-based security option on the configuration page for outgoing webhooks by selecting the corresponding checkbox. Enter a secret key consisting of alphanumeric characters (lowercase letters a-z, digits 0-9) and is 32 characters.

 - **Signature-based security:** The receiver requires the server and client authentication tokens to be equal.

When an event of interest is triggered, a webhook payload is sent to your application with the `X-OCM-Webhook-Signature` header in the signed event. The `X-OCM-Webhook-Signature` header includes an epoch timestamp and a signature, for example:

```
{
    X-OCM-Webhook-Signature:
        t=1660807672,
    v=d1b1ea85b53f8dd435a96f59d778d9cdb09be1905d1cecc7acdf4e03ecf60a1b
}
```

The value of `t` is an epoch timestamp, computed at the time of sending the request and is used in computing the signature `v` when sending the request.

The value of `v` is a signature, computed from `t` and the payload, i.e. `hash(t + "." + payload)`

Upon receiving the webhook request, verify the signature:

- a. Obtain the values of the epoch timestamp `t` and the signature `v` by parsing the request header `X-OCM-Webhook-Signature`.
- b. Compute the current epoch timestamp `t1` and ensure that the difference between `t1` and `t` (epoch timestamp obtained from step a) is not more than the threshold, such as 30 or 60 seconds. Go to the next steps only if the difference between the epoch timestamps is less than the threshold.
- c. Use the value of timestamp `t` from step a and the payload from the request to create a string by concatenating as follows:
`t + "." + payload`
- d. Using the secret key from the webhook configuration page when you opted for the signature-based security, compute the HMAC signature of the string created in step c with SHA-256 algorithm. Ensure that the signature is encoded in hex before proceeding to the next step. The `javax.crypto.Mac` (part of `jdk-1.8` library) can be used to generate the HMAC signature. Additionally, a third-party library such as [Bouncy Castle](#) can also be used to generate the signature for the payload.
- e. Compare the signature generated in step d with the signature from the request header. Continue with further processing of the request only if the signature values match.

14. Click **Save**.

To delete a webhook, click **Delete** next to the webhook.

To edit a webhook, click **Edit** next to the webhook.

Monitor Webhook Events

As an administrator or developer, you can access a log of events posted to a Webhook. The **Events Log** displays the Object IDs, Event IDs, and Published dates and times for webhook activities in an Oracle Content Management instance.

To monitor Webhook events:

1. After you sign in to the Oracle Content Management web application as an administrator or developer, click **Integrations** in the **Administration** area of the navigation menu on the left.

- Click **Webhooks** in the Integrations dropdown menu.
- On the settings page for a webhook instance, choose the option to open the **Events log** page.
The **Events log** page displays a list of all the events published to this webhook. Recent posts are shown at the top.

Test Webhook		
General	Event Log	
Object/ID	Event ID	Published On
Test Webhook 1524	763b1e87-afa9-4980-a5e0-f6d6804b40a7	3/9/2020 at 2:58 PM
Test Webhook 1521	10863de2-cc81-42e4-8f8a-7a23af9696b5	3/9/2020 at 2:55 PM
Test Webhook 1505	8c22bf90-e89c-4748-b332-fc93adcdfa60	3/9/2020 at 2:54 PM
Test Webhook 1491	763b1e87-afa9-4980-a5e0-f6d6804b40a7	3/9/2020 at 2:52 PM
Test Webhook 1476	8c22bf90-e89c-4748-b332-fc93adcdfa60	3/9/2020 at 2:50 PM
Test Webhook 1483	10863de2-cc81-42e4-8f8a-7a23af9696b5	3/9/2020 at 2:49 PM
Test Webhook 1456	763b1e87-afa9-4980-a5e0-f6d6804b40a7	3/9/2020 at 2:45 PM
Test Webhook 1440	8c22bf90-e89c-4748-b332-fc93adcdfa60	3/9/2020 at 2:45 PM
Test Webhook 1448	10863de2-cc81-42e4-8f8a-7a23af9696b5	3/9/2020 at 2:42 PM
Test Webhook 1422	b1a783f8-dd66-4c62-b9e9-1b373060e1fd	3/9/2020 at 2:41 PM
Test Webhook 1412	e635cf15-3815-4953-9e4c-6db083436602	3/9/2020 at 2:41 PM

For each event, the log also shows its response status (**Success** or **Failure**). You can expand a posted event to see details about notifications sent to a webhooks client.

The following table shows the details of the events that are triggered:

Event	Details
SCHEDULEDPUBLISHING_PARENTCREATED - create a recurring scheduled publishing job	<p>Key: cec.event.scheduledpublishing.parentcreated.name</p> <p>Display Name: Scheduled publishing job created</p> <p>Message Detail: Scheduled publishing recurring job '{job-name}' '{job-id}' created at '{datetime}'</p>
SCHEDULEDPUBLISHING_PARENTUPDATED - properties of the whole recurring job is updated (for example, the series name or description is updated, job manager is added or removed)	<p>Key: cec.event.scheduledpublishing.parentupdated.name</p> <p>Display Name: Scheduled publishing job updated</p> <p>Message Detail: Scheduled publishing recurring job '{job-name}' '{job-id}' updated at '{datetime}'</p>
SCHEDULEDPUBLISHING_CREATED - create a scheduled publishing job	<p>Key: cec.event.scheduledpublishing.created.name</p> <p>Display Name: Scheduled publishing job created</p> <p>Message Detail: Scheduled publishing job '{job-name}' '{job-id}' with assets '{the-first-10-asset-guids}' created at '{datetime}'</p>
SCHEDULEDPUBLISHING_UPDATED - update a scheduled publishing job	<p>Key: cec.event.scheduledpublishing.updated.name</p> <p>Display Name: Scheduled publishing job updated</p> <p>Message Detail: Scheduled publishing job '{job-name}' '{job-id}' with assets '{the-first-10-asset-guids}' updated at '{datetime}'</p>
SCHEDULEDPUBLISHING_CANCELLED - cancel a scheduled publishing job	<p>Key: cec.event.scheduledpublishing.cancelled.name</p> <p>Display Name: Scheduled publishing job cancelled</p> <p>Message Detail: Scheduled publishing job '{job-name}' '{job-id}' with assets '{the-first-10-asset-guids}' cancelled at '{datetime}'</p>
SCHEDULEDPUBLISHING_STARTED - scheduled publishing job is started	<p>Key: cec.event.scheduledpublishing.started.name</p> <p>Display Name: Scheduled publishing job started</p> <p>Message Detail: Scheduled publishing job '{job-name}' '{job-id}' with assets '{the-first-10-asset-guids}' started at '{datetime}'</p>
SCHEDULEDPUBLISHING_COMPLETED - scheduled publishing job is completed	<p>Key: cec.event.scheduledpublishing.completed.name</p> <p>Display Name: Scheduled publishing job completed</p> <p>Message Detail: Scheduled publishing job '{job-name}' '{job-id}' with assets '{the-first-10-asset-guids}' completed at '{datetime}'</p> <p>A failed reason will be appended if the job is failed.</p> <p>Message Detail: Scheduled publishing job '{job-name}' '{job-id}' with assets '{the-first-10-asset-guids}' completed at '{datetime}'. {if-failed-failed-reason}</p>

The Oracle Cloud [REST API for Activity Log](#) provides the ability to search activities in Oracle Content Management. This API has the following endpoints:

- **Audit Log**, which provides the details of activities and related data.

- **Events**, which provides the details of types and categories.

Use Endpoints for Push Notifications

Oracle Content Management automatically delivers notifications to webhook endpoints about asset publishing, asset lifecycle, site publishing, prerendering, and scheduled jobs events.

You subscribe to events for your webhooks to receive notification when the events happen in the payloads for the endpoints. For example, if your endpoint subscribes to an event called `DIGITALASSET_CREATED`, when a digital asset is created in a repository you specified, the webhook payload can tell you the name of the webhook, what time the event happened, and who did the event.

You can use an endpoint to be notified when some action happens. Then Oracle Content Management calls that endpoint. You can create webhooks using REST endpoints, similar to other REST APIs. You need to specify a server URL for each webhook endpoint. The GET, POST, PUT, and DELETE endpoints are hosted on the Oracle Content Management server, so the same host name, port, and system is the context for all the webhook endpoints.

You can create all the content items with the REST API headless endpoints. You can also create webhooks using the REST API, and listen to the events.

The webhook endpoints are as follows:

GET /webhooks	Lists all webhooks.
POST /webhooks	Creates a webhook with the given information in the payload.
GET /webhooks/{id}	Reads a webhook with the given ID.
PUT /webhooks/{id}	Updates a webhook with the given information in the payload.
DELETE /webhooks/{id}	Deletes a webhook with the given ID.

Or you can create, configure, enable, or disable webhooks in the Oracle Content Management web UI at **Administration > Integrations > Webhooks**. See [Configure Webhooks](#).

You can view event payloads, with **Brief** or **Detailed** output. Webhook payloads contain the following data:

- Which webhook got invoked
- What event got triggered (maybe `CONTENTITEM_UPDATED` or `CHANNEL_ASSETPUBLISHED`)
- When the event was registered, and which user triggered it
- What entity is the subject of the event:
 - For asset lifecycle webhooks, it can be a content item or digital asset.
 - For asset publishing events, it will be a list of the identifiers of all published content.

Webhooks have three different formats, as the following table describes.

Format	Output
Brief	Output includes only basic information in the payload. The output gives you details like what the action is, when it happened, in what repository, and who did the action.
Detailed	Output includes all content information in the payload. The output gives you the whole content item data, in a URL that you specify.
Empty	Available only for asset publishing webhooks. If you specify Empty for the output, the payload contains only an identifier representing the publish session ID.

If the endpoint requires authentication, you can select what type of authentication and then enter the authentication information.

Receive Push Notifications from an Asset Lifecycle Webhook

An asset lifecycle webhook can send you push notifications about asset lifecycle events in an Oracle Content Management repository for content items and taxonomies.

You can receive information notifications about the following events for your webhooks:

Events

All current and future events

Select individual events

General Type	Created	Updated	Deleted	Approved	Published	Republished	Unpublished
Content Item	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Digital Asset	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Collection	<input type="checkbox"/>		<input type="checkbox"/>				

Asset Translation Type	Language Added	Translated	Completed
Content Item	<input type="checkbox"/>	<input type="checkbox"/>	
Digital Asset	<input type="checkbox"/>	<input type="checkbox"/>	
Translation Set			<input type="checkbox"/>

Asset Categorization Type	Category Added	Category Removed
Content Item	<input type="checkbox"/>	<input type="checkbox"/>
Digital Asset	<input type="checkbox"/>	<input type="checkbox"/>

Asset Archiving Type	Archived	Restored
Content Item	<input type="checkbox"/>	<input type="checkbox"/>
Digital Asset	<input type="checkbox"/>	<input type="checkbox"/>

A repository manager can create an asset lifecycle webhook with a REST API endpoint or on the **Administration > Integrations > Webhooks** page in the Oracle Content Management web interface. On the page for a webhook, users can configure the following settings:

- **Name:** A webhook name.

- **Description:** A webhook description.
- **Type:** A webhook type.
- **Enable Webhook:** An **enabled** or **disabled** status for the webhook. The default is **disabled**.
- **Repository:** A repository for setting the event scope for the webhook notifications.
- **Asset Types:** A list of asset types for setting the event scope of the webhook notifications. The filtering feature applies only to events published for content items or digital assets of chosen types.
 - Use the **All asset types** option to receive notifications about lifecycle events for assets of any type that is assigned or will be assigned to the repository(s).
 - Use the **Select individual asset types** option to receive notifications about lifecycle events for assets of chosen types only. As an administrator, you can select both content and digital asset types. Note that only asset types that are assigned to the selected repository(s) are displayed in the asset type picker.
- **Events:** A list of events for the webhook to receive notifications. Use the **All current and future events** option for the webhook to receive notifications for all events. Use the **Select individual events** option for the webhook to receive notifications for specific events.
- **Payload:** The type of payload to send to the endpoint. (**Brief** or **Detailed**)
- **Target URL:** The URL of the endpoint to which notifications should be posted.
- **Authentication:** A callback receiver security (**None**, **Basic**, **Header**)
- **Self-signed SSL certificate:** The receiver accepts a self-signed SSL certificate.
- **Signature-based security:** The receiver requires the server and client authentication tokens to be equal.

Name, **Description**, **Enable Webhook**, **Target URL**, and **Authentication** are standard settings for a webhook. For the repository setting, pick a repository for which you have Manager permission.

Receive Push Notifications from an Asset Publishing Webhook

An asset publishing webhook can send you push notifications for asset and taxonomy publishing events in Oracle Content Management.

You can receive information about the following events from an asset publishing webhook:

Type	Published	Unpublished
Channel Asset	<input type="checkbox"/>	<input type="checkbox"/>
Channel Taxonomy	<input type="checkbox"/>	<input type="checkbox"/>

A channel manager can create a new asset publishing webhook with a REST API endpoint or on the **Administration > Integrations > Webhooks** settings page in the Oracle Content Management web interface. This enables notifications about assets published to the following channels:

- Public or secure custom publishing channels
- Publishing channels that Oracle Content Management creates for public or secure enterprise sites

You can receive notifications about the following activities in the scope of a channel:

- Published: A new content item, digital asset, or taxonomy is published.
- Unpublished: A content item, digital asset, or taxonomy is unpublished.

Receive Push Notifications from a Site Publishing Webhook

A site publishing webhook can send you push notifications for site publishing events in Oracle Content Management.

A site manager can create a new site publishing webhook with a REST API endpoint or on the **Administration > Integrations > Webhooks** settings page in the Oracle Content Management web interface. This enables notifications for site events with types including published, status, and unpublished.

Receive Push Notifications from a Prerender Webhook

A prerender webhook can send you push notifications for prerender events in Oracle Content Management.

A site manager can create a new prerender webhook with a REST API endpoint or on the **Administration > Integrations > Webhooks** settings page in the Oracle Content Management web interface.

This outgoing webhook for the prerenderer enables you to set up a notification about a prerender event, which enhances the prerender server to allow prerendering and caching of a Detail page after an asset is published to the site's channel.

As an administrator, you can create a Site Detail Page Refresh Webhook to notify Prerender about the need to render and cache a Detail page.

Receive Push Notifications from a Scheduled Jobs Webhook

A scheduled jobs webhook can send you push notifications for publishing jobs scheduled in a repository in Oracle Content Management.

A site manager can create a new scheduled jobs webhook with a REST API endpoint or on the **Administration > Integrations > Webhooks** settings page in the Oracle Content Management web interface. This enables notifications for scheduled publishing jobs in a repository.

You can receive notifications for scheduled job events with types including created, updated, canceled, started, and completed.

Incoming Webhooks

You can use webhooks to have Oracle Content Management automatically receive a notification based on various events happening in the external services.

The incoming webhooks framework provides a way with which external applications or services can call and notify Oracle Content Management about various events that are

required by Oracle Content Management. The webhook framework provides a means for validating the incoming request with token-based authentication.

**Note:**

Incoming webhooks aren't supported in private instances.

1. After you sign in to the Oracle Content Management web application as an administrator or developer, click **Integrations** in the **Administration** area of the navigation menu on the left.
2. The Oracle Content Management components that need webhook support will create an incoming webhook instance. For example, when you create a Lingotek translation connector, as [Request a Lingotek Trial Connector for Content Translation](#) describes, the connector framework in the background creates an instance of the incoming webhook that gets associated with that connector. In the **Integrations** menu, click **Webhooks** to see the list of webhook instances of Lingotek translation connectors.
3. On creating a webhook instance, the framework generates a security token with a configurable duration of validity. Every incoming request should have this token. The framework validates the incoming request by verifying the token and its expiry. The component submits a callback URL and token to the external application. When any event in which the component has subscribed to the external application occurs, the component makes a call back to Oracle Content Management over the same URL.

Click the checkbox next to your incoming webhook instance, and then select **Edit** from the action toolbar. Or, you can click on the clickable name of the incoming webhook to get to the **Edit** page. On the **General** page, you can specify the values for **Service Valid For** and **Security**. **Name** and **Description** are populated at the time the incoming webhook is created, but you can change these values if you want to. The **Target URL** field is read-only. The server generates the URL when the webhook instance is created.

After clicking the checkbox you'll need to select **Edit** from the action toolbar. Or, you can click on the clickable name of the incoming webhook to get to the **Edit** page.

4. The **Event Log** page displays a list of all the events. The list displays the login ID, date and time when the event occurred, status (success or failure), and download icon (to download the event log).

Configure an Incoming Webhook

Use the Oracle Content Management web interface to configure adapter fields and the CAL adapter for an incoming Webhook.

To configure a Lingotek Incoming Webhook:

1. After you sign in to the Oracle Content Management web application as an administrator, click **Integrations** in the **Administration** area of the navigation menu on the left.
2. In the **Integrations** menu, click **Webhooks**.
3. Fill in the following fields for the Lingotek-Incoming Webhook.

The **Name** and **Description** fields are populated at the time the Incoming Webhook is created. You can edit and change these values if you want to.

The screenshot displays the 'Content Management' interface for configuring an incoming webhook. The left sidebar contains navigation options: Home, Assets, Sites, Experiences, Recommendations, Developer, Analytics, COLLABORATION (Documents, Conversations, Sauce Video), ADMINISTRATION (System, Integrations), and Content. The main content area shows the configuration for a webhook named 'Lingotek Connector-IncomingWebhook...'. The 'General' tab is selected, and the 'Event Log' tab is also visible. The configuration fields are as follows:

- Name***: Lingotek Connector-IncomingWebhook-
- Description**: A generic webhook for Lingotek Connector
- Type**: Generic Connector
- Enable Webhook**:
- Service Valid For**: [] []
- Target URL***: []
- Security**: Token [Details]
- Include Request Headers**:
- Include Request Parameters**:
- Include Body**:
- Webhook Scope**: Enter Webhook Scope

Use the CAL-Based Webhook Adapter

The CAL-based webhook adapter publishes the details of an incoming request on the CAL network.

Any interested component can consume the event and further process it. A component that needs to consume the webhook can implement the processor.

Set Proxies

To set proxies, you can configure a proxy server that provides a URL to which web browsers can connect.

- [Configure Proxy Service Settings](#)
- [Add Logged User Data to a Request Through a Secure Proxy Endpoint](#)
- [Debug Proxy Service Endpoints](#)

Configure Proxy Service Settings

Oracle Content Management includes a proxy service, so that you can use REST services that have Cross-Origin Resource Sharing (CORS) limitations or require service account credentials.

The proxy service is a reverse proxy server. It provides a URL to which web browsers connect. The proxy service then acts as an intermediary between the web browser and a remote REST service (or endpoint). The proxy service explicitly adds CORS support to all endpoints and can optionally insert service account credentials to requests coming from web browsers.

If you are using a REST server (or endpoint) that already supports CORS and doesn't require the use of service account credentials, you don't need to register it with the proxy service. You can instead register it directly with the Oracle Cloud REST API for Content Management.

1. After you sign in to the Oracle Content Management web application as an administrator, click **Integrations** in the Administration area of the navigation menu.
2. Under **Proxy Service**, select **Enable**.
3. Using the following steps, define any credentials needed by your endpoints, and define the endpoints you want to use the proxy service.

Credentials

When an endpoint uses a credential, the proxy service adds basic access authentication (via the HTTP Authorization header) to requests made by web browsers. If the browser request already includes the Authorization header, the browser request Authorization header will be used instead of the one in the credential.

This gives you the flexibility to provide a read-only credential for most requests, but allow individual requests to provide their own write-capable authentication as needed.

Providing a credential to an endpoint gives all users of the endpoint the same effective permissions granted to the user defined in that credential. To ensure you don't inadvertently create a security risk, take the following precautions:

- Don't provide a credential for an endpoint unless absolutely necessary. If possible, let the browser requests provide their own Authorization header instead.
- If you must provide a credential, try to use one which has read-only access on the target endpoint.

- Limit the allowed methods on the endpoint to what is actually required. Unless absolutely necessary, always disable the PUT, POST, and DELETE methods on an endpoint.
- When possible, limit the Target URI for the endpoint to a specific area of functionality. For example, rather than providing the base URI to the full API such as `http://example.api/`, you might be able to limit it to a specific area such as `http://example.api/weather/` (for weather-related requests) or `http://example.api/date/` (for date-related requests).

If an endpoint requires credentials, define a credential and select it in the endpoint definition:

1. Click **Create new Credential**, and complete the following information.
2. In the **Credential Name** box, enter a name for the credential that will make clear to other users what the credential is for (for example, `DocsAPIUser`).
3. In the **Username** box, enter the user that should be used to authenticate all requests with the endpoint.
4. In the **Password** box, enter the password for the user you entered.
5. In the **Keywords** box, optionally provide space-delimited keywords for the credential. Keywords are purely informational for your own needs and do not alter the functionality of the credential. Keywords can include alphanumeric characters, periods, hyphens, and underscores.

The **Keywords** field is exposed by the proxy service API and can be viewed by non-administrator users. Never include user names, passwords, API keys, or other sensitive information in the **Keywords** field.

6. Click **Save**.

The new credential is available to use with one or more endpoints. It appears in the **Credential** drop-down list when you create or edit an endpoint.

Endpoints

1. Define the remote API endpoint you want to use the proxy service. Click **Create new Endpoint**, and complete the following information:
 - a. In the Endpoint Name box, enter a name for the endpoint that will make clear to other users what this endpoint is (for example, `Content Management API 1.1`).
 - b. Under **Enable Endpoint**, select **Enabled**.

You can disable individual endpoints as necessary, rather than disabling the whole proxy service.
 - c. Under **Secure Endpoint**, select **Secure Endpoint**. You can secure an endpoint by allowing only authenticated user to access the endpoint. If you select this check box, anonymous users are required to authenticate themselves via a sign-in page.
 - d. In the **Path Name** box, enter a path name for the endpoint (for example, `docs`). This will become part of the URL path to access the endpoint (for example, `/pxysvc/proxy/docs`).

The name must be unique, URL-safe, and lowercase, and it must start with a letter. It can include alphanumeric characters, hyphens, and underscores.

- e. In the **Target URI** box, enter the URI for the endpoint (for example, `http://service.example.com/documents/api/1.1`).
- f. Under **Credential**, if necessary, select the credentials to use for this endpoint. This list is populated by the credentials you created using the steps above.
- g. Under **HTTP Methods**, select the HTTP methods you want to enable for this endpoint.
GET and OPTIONS methods are always enabled.
- h. In the **Keywords** box, optionally provide space-delimited keywords for the endpoint. Keywords are purely informational for your own needs and do not alter the functionality of the endpoint. Keywords can include alphanumeric characters, periods, hyphens, and underscores.

The **Keywords** field is exposed by the proxy service API and can be viewed by non-administrator users. Never include user names, passwords, API keys, or other sensitive information in the **Keywords** field.
- i. Under **Additional Headers**, you can define multiple custom headers to be sent to the target endpoint. For example:

```
CustomHeader1: Value1
CustomHeader2: Value2
```

The following example from a debug interface shows all the headers sent to a target:

```
ids_cloudgate_id: internal_APPID
ids_client_id: internal_APPID
CustomHeader1: Value1
CustomHeader2: Value2
Authorization: *****
```

The proxy service will add these headers for each request sent to target URI. They will be part of every request header sent to target URI when a proxy URL is requested.

- j. In the **Connection Timeout** box, enter the maximum number of seconds to wait when trying to make a connection with the target URI.
- k. In the **Socket Timeout** box, enter the maximum number of seconds to wait for a pooled connection in the proxy service.
- l. In the **Connection Request Timeout** box, enter the maximum number of seconds to wait when trying to make a connection with the proxy service.
- m. Test your endpoint, by clicking **Save** and **Debug**. See [Debug Proxy Service Endpoints](#).
- n. When you're satisfied with the result, click **Save** and **Close**.

Add Logged User Data to a Request Through a Secure Proxy Endpoint

You can add user data to an external service call with the Oracle Content Management Proxy.


As a system administrator, you can add user profile information including user ID, full name, and email to an external service call with the Oracle Content Management Proxy. You can secure an endpoint only for authenticated users and proxy automatically passes their profile information to the external system. The external system can then assume that the users are authenticated, and it accesses other systems on their behalf.

An anonymous user is navigated to the sign-in page for authentication if [a secured endpoint is configured](#).

Debug Proxy Service Endpoints

You can quickly test a proxy service endpoint without writing any test code. This can be a valuable tool to see how requests and responses are handled between your web browser, the proxy service, and the target URI of the endpoint.

1. After defining the endpoint, test it by clicking **Save and Debug**.

Alternatively, in the list of endpoints, click  next to the endpoint you want to debug.

At the top of the **Debug Endpoint** section, you see the **URI mapping**, which shows how the endpoint's local path name is mapped to the target URI of the endpoint. Both the local path and the target URI are links so you can make sure they are pointing to the correct locations.

2. Optionally, enter a user name and password and additional headers to use for the test request. These credentials will be used in lieu of any credential already set on the endpoint. This enables you to test or verify the use of credentials without making them available to all users of the endpoint.
3. Optionally, enter additional headers to use for the test request. Enter one header per line, using the standard format *Name: Value*. For example: `Content-Type: application/json`.
4. Select an HTTP method, then provide a complete path for the request.
If you select **POST** or **PUT**, in the **Data** box, enter the content which should be sent in the POST or PUT request body.
5. Click **Submit Request**.
6. In the **Debug Result** section, expand the panels to see the detailed results:
 - **Browser Request to Proxy:** Displays the HTTP request headers and body sent by the web browser to the proxy service.
 - **Proxy Request to Endpoint:** Displays the HTTP request headers and body sent by the proxy service to the target URI of the endpoint. If the endpoint uses a credential, an authorization header is inserted in the request, but the header value isn't shown in the debug results.
 - **Endpoint Response to Proxy:** Displays the HTTP response headers and body sent by the target URI of the endpoint back to the proxy service.
 - **Proxy Response to the Browser:** Displays the HTTP response headers sent by the proxy service back to the web browser. If the target URI of the endpoint doesn't return CORS headers, the proxy service inserts them in its response to the browser.
7. If necessary, change the debug request and submit the request again.

 **Note:**

The old debug results are overwritten with the new results.

8. When you're satisfied with the result, click **Close**.